TOPOLOGICAL STRUCTURES FOR GENERALIZED BOUNDARY REPRESENTATIONS

Leonidas Bardis and Nicholas Patrikalakis

MITSG 94-22

Sea Grant College Program
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Grant: NA90AA-D-SG424
Project No: 92-RU-23

Related MIT Sea Grant College Program Publications

Feature extraction from B-spline marine propeller representations. Nicholas M. Patrikalakis and Leonidas Bardis. MITSG 93-12J. $2.

Cut locus and medial axis in global shape interrogation and representation. Franz-Erich Wolter. MITSG 93-11. $4.

An Automatic coarse and fine surface mesh generation scheme based on medial axis transform. Part I, Algorithms. Part II, Implementation. H. Nebi Gursoy and Nicholas M. Patrikalakis. MITSG 93-7J. $2.

Offsets of curves on rational B-spline surfaces and surface approximation with rational B-splines. L. Bardis and N. M. Patrikalakis. MITSG 91-24J. $4.

Surface intersections for geometric modeling. N. M. Patrikalakis and P.V. Prakash. MITSG 91-23J. $2.

Please add $1.50 ($3.50 foreign) for shipping/handling and mail your check to :
MIT Sea Grant College Program, 292 Main Street, E38-300, Cambridge, MA 02139.

Topological Structures for Generalized Boundary Representations

Leonidas Bardis

Nicholas M. Patrikalakis

Massachusetts Institute of Technology
Cambridge, MA 02139-4307, USA

Design Laboratory Memorandum 91-18
Issued: September 7, 1991
Revised: June 4, 1992
Revised: July 2, 1992
Revised: September 24, 1993
Revised: January 12, 1994
Revised: September 13, 1994

Dedication

To our families.

Contents

List of Figures

Preface

Solid models have become an essential constituent of modern CAD systems. Boundary representation models are the natural choice when the solid object is bounded by free–form curves and surfaces. There exist a great number of solid modelers based on manifold solid representation, which use Euler operators for incremental object construction. Development of models for representation of non–manifold objects has become an active research topic. On the other side, some new data structures for genuine manifold representation have been devised, which are based on topological and graph theoretical concepts and do not rely on Euler operators. This monograph reviews the most well known models of the above three kinds. For each modeler reviewed here, the representation space, data structures and basic constructors are discussed. Essential concepts from topology and graph theory are treated briefly. Special emphasis has been placed on the cell–tuple structure due to its expressive power and elegant mathematical formulation. Finally, this monograph provides an overview of applications of geometric modeling in Computer–Aided Engineering and specifically discusses the relative merits of various boundary representation modeling systems in this context.

The authors are indebted to Prof. C. Chryssostomidis for his assistance and valuable comments. Early discussions with Dr. C. Bliek and Dr. W. Hansmann on solid modeling issues are also acknowledged. Useful detailed comments on an earlier draft of this monograph were provided by Dr. E. Brisson, Mr. C.–Y. Hu, Dr. J. R. Rossignac, Dr. N. Sapidis, Mr. E. C. Sherbrooke, and Prof. F.–E. Wolter and are greatly appreciated. Mr. S. L. Abrams and Mr. C.-Y. Hu assisted in editing the monograph.

Funding for this work was obtained in part from the Department of Commerce via the MIT Sea Grant College Program under grant NA90AA-D-SG424, the Office of Naval Research under grant N00014-91-J-1014, and the National Science Foundation under grant numbers IRI-9224640 and DDM-9215411.

L. Bardis
N. M. Patrikalakis
Massachusetts Institute of Technology
Cambridge, MA, USA

Chapter 1

Introduction

Representation of solid objects in an abstract form permitting storage, evaluation and processing by a computer is the primary objective of *solid modeling*. During the past 15 years extensive research has been carried out in this area and solid modeling systems have emerged with a wide variety of representation capabilities. Engineering applications of solid modeling techniques were aimed mostly at design, analysis and manufacturing of mechanical machine parts and assemblies. On the other hand, no significant development has taken place in design and analysis of marine and aeronautical structures bounded by sculptured surfaces and involving an extremely complex internal subdivision. This monograph attempts a comparative study of existing solid modeling methods in terms of their data structure and modeling space.

Point set topology [4, 48, 74, 103], algebraic topology [1, 39, 71, 75], and graph theory [8, 45] provide theoretical foundations for modern solid modeling. Three major representation models have emerged: decomposition, constructive and boundary models [26, 49, 69]. Decomposition models are based on a limited class of primitive objects for the description of point sets. Quadtrees and octtrees are the most successful models of this category. Quadtrees, [53, 97] are a decomposition of a two-dimensional point set into an arrangement of squares or rectangles covering the set. Initially, a minimal bounding rectangle is computed such that the point set S is a subset of the point set represented by the rectangle. This bounding rectangle occupies the root of the representation tree. Then, this is subdivided into four equal rectangles, R_i, $i = 1, 2, 3, 4$, represented by four branches starting from the root node. The nodes at the lower end of these branches are classified according to the set theoretic relations of the point sets represented by the respective rectangles and the original point set. The tree node is black if $R_i \subseteq S$, white if $R_i \cap S = \emptyset$, and grey if $R_i \cap S \neq \emptyset$ and neither of $R_i - S$ and $S - R_i$ is empty. The above subdivision process is repeated at grey nodes until all nodes are either black or white or the rectangle dimensions are smaller than a prespecified tolerance. Quadtrees owe their name to the 4-ary tree structure used for the representation.

Octtrees [73, 99, 98, 119] are the three-dimensional counterparts of quadtrees. The primitive used for three-dimensional point set subdivision is usually a cube. The point set classification step after subdivision of a cube involves eight octants, which now require an 8-ary tree as a data model, hence the name octtree. Quadtrees and octtrees have been used in image processing, rendering, surface and volume calculations. Data structures for their representation and algorithms for their

interrogation (calculation of geometric properties, point classification, geometric transformations, ray tracing, medial axis transforms) are discussed in [99, 98]. Octtrees require an excessive amount of data in order to achieve an accurate surface representation compared with other representation models. The storage complexity is of the order of the surface area of the objects. Further, the octtree subdivision is dependent on the position and orientation of the object with respect to the coordinate system used. The construction of an octtree subdivision is generally based on some alternate representation or measured data. In [16, 100] some general spatial data structures based on the octtree approach for representing solids with curved faces are introduced.

Constructive models describe point sets with the help of a number of primitive point sets. In half-space models, these primitives are defined by algebraic inequalities. In models using the Constructive Solid Geometry (CSG) approach [86, 90, 88] the primitives are geometric figures having a relatively simple shape, such as cubes, cylinders, spheres, torii, etc. A solid is represented as the root of a binary tree, whose nodes are Boolean set operators and leaves are occupied by instantiated primitives of the above type. Instantiation involves specification of size, position and orientation of a primitive. Set operators include union, intersection and difference of point sets. In some pathological cases, solids containing two-dimensional or even one-dimensional components may result from the above set operations, such as dangling edges or faces. To permit a closed representation, regularized set operations have been introduced [89]. The result of a regularized set operation is the closure of the interior of the set obtained from the mathematical definition of the set operation involved. CSG models are easy to use and cover a wide variety of solid shapes, especially mechanical parts and assemblies. However, interrogation of CSG models requires their conversion to decomposition [61, 63, 64] or boundary models [10, 15]. In some cases, special techniques, such as ray casting are used [96]. Further, CSG is not particularly well suited to model thin shell structures and manufacturing procedures common to marine and aerospace structures such as heating, welding, and rolling [26].

Boundary models describe solids in terms of their bounding entities, such as faces, loops, edges and vertices. In Boundary representation (B-rep) there is a clear separation between topological and geometrical data. Topological data include connectivity information, i.e. adjacency relations between zero-dimensional (vertices), one-dimensional (edges), two-dimensional (faces) and three-dimensional (shells) entities. Description of entities of the above type through mathematical curves and surfaces, such as straight lines, rational B-spline curves, planes, quadrics, rational B-spline surface patches, is the basis of the geometrical data base of a B-rep model. Topological information is highly structured, whereas geometrical information has the form of a data repository pointed to by abstract topological entities. Most B-rep modelers are based on planar, polyhedral geometry [7, 11, 43, 70] although systems using free-form surfaces as geometrical primitives have also started to appear [23].

Abstract models for the representation of topological information in B-rep modelers are subsets of incidence graphs implemented by pointer structures. One major requirement of the data structure is topological sufficiency, i.e. reconstruction of all adjacency relations involving any topological entity. For example, given a particular face, find all edges and vertices of its boundary. Space complexity of the underlying data structure and time complexity of interrogation algorithms play an important role in the design and implementation of a B-rep modeler. Another important characteristic of a modeling system based on B-rep representation is the ability to handle non-manifold and mixed-dimension structures, such as polyhedra with dangling edges or faces, solids

intersecting each other along an edge or touching on a single vertex, etc. [43, 94, 112].

Parallel research directions in geometric modeling based on a B-rep representation resulted in three model categories. Models of the first category deal with manifold structures and use Euler operators as a structural primitive. The topological structure corresponding to an object is created by adding or removing a minimum number of entities each time while preserving the validity of the Euler-Poincaré formula. High level operators are implemented on top of Euler operators and are used to build a complex structure, such as building a polyhedral model of a quadric, joining or splitting solids along a common face, sweep and rotational operators and Boolean operators [7, 11, 23, 70, 108].

Models of the second category use additional topological entities and relations to represent non-manifold, mixed-dimensional structures. Euler-like or other primitive operators are used to build models incrementally. Two of the known models of this type handle non-manifold, three-dimensional objects [43, 112, 110, 109], while recently a model for non-manifold n-dimensional objects has appeared [94].

Models of the third category are based on abstract mathematical structures for the representation of manifold subdivisions of two, three, or n-dimensional space. Such models utilize a small number of low level constructors maintaining the validity of the basic topological structures [12, 13, 14, 29, 30, 40, 65, 66]. High level operators may be implemented based on the low level ones as in models of the other categories. Such models have been used primarily for the representation of the solution space of computational geometry problems, such as calculation of Voronoi diagrams and Delaunay triangulations. Nevertheless, they provide powerful modeling tools for scientific and engineering applications.

The above classification of B-rep models is based on one side on the modeling space and on procedures followed to build complex objects on the other side. Other classifications using other criteria are possible, such as internal characteristics of the data structure or representation of ordering information [66].

In general, B-rep modelers provide a suitable basis for the implementation of interrogation algorithms, such as graphical rendering, Boolean operations, calculations of integral properties (area, volume, center of volume, moments of inertia, etc.). On the other hand, they are sensitive to numerical inaccuracies, [50, 49]. Hybrid modelers [90, 88] use the CSG representation as a user interface and algorithms for converting the CSG to a B-rep model. Conversion algorithms make heavy use of intersection calculations between surfaces, since a CSG model is based on Boolean operations on solid primitives bounded by curved surfaces [18, 37, 57, 58, 60, 59, 84].

In [91] a brief account of recent developments and outstanding problems in solid modeling is given.

This monograph is structured as follows. Chapter 2 provides a summary of mathematical definitions and theorems, which are necessary to enhance clarity of the rest of the monograph. Chapter 3 is a review of systems based on the boundary representation method. Chapter 4 contains some conclusions and provides some recommendations on possible applications of existing theory and algorithms in the design, analysis and manufacture of structures bounded by sculptured surfaces and involving extremely complex architectural subdivisions and structural features.

Chapter 2

Mathematical Background

As is mentioned in the Introduction, there is a clear distinction between topological and geometrical information in B-rep models. Topology deals with properties of point sets that are invariant under continuous geometric transformations. We repeat here some basic definitions and theorems from topology [4, 48, 49, 74, 103], algebraic topology [1, 71, 75], and graph theory [8, 45].

2.1 Topological Concepts

Definition 1 A topological space (X, T) is a pair composed of a set X and a set T of subsets of X, the *open* sets of the topological space (X, T) or simply *open* sets of X, such that

1. the intersection of finitely many sets in T is again in T and
2. the union of sets in T is also in T.

In view of the above definition, open sets are introduced in an axiomatic manner in contrast to the approach followed in modern analysis, where open sets are defined with the help of a metric. It should be noted that the union of an infinite number of open sets is also an open set.

Definition 2 A set P is closed if $X - P$ is an open set.

The Euclidean space $\Re^n = (x_1, x_2, \ldots x_n)$, $x_i \in \Re$, $1 \leq i \leq n$ with $T = \left\{\vec{B}^n(\vec{p}, r),\ r > 0,\ \vec{p} \in \Re^n\right\}$ is a topological space. $\vec{B}^n(\vec{p}, r)$ is defined as follows.

Definition 3 The open n-ball $\vec{B}^n(\vec{p}, r)$ of radius r centered at $\vec{p}$ is

$$\vec{B}^n(\vec{p}, r) = \{\vec{q} \in \Re^n,\ ||\vec{p} - \vec{q}|| < r\} \tag{2.1}$$

where $||\ ||$ is the Euclidean norm of a vector in $\Re^n$

$$\vec{x} = (x_1, x_2, \ldots, x_n),\ ||\vec{x}|| = (\sum_{i=1}^{n} x_i^2)^{1/2} \tag{2.2}$$

Definition 4 A neighborhood of a point $\vec{x} \in X$ is an open set $U \subseteq X$ containing $\vec{x}$. It can be proven that a set is open if it contains a neighborhood for each of its points.

Definition 5 A topological space (X, T) is called a Hausdorff space, if there exist disjoint neighborhoods for each pair of distinct points of the set X.

Definition 6 The interior $int(P)$ of any set $P \subseteq X$ is defined as the set of all points of P which have a neighborhood lying entirely in P.

Definition 7 The closure of P, $cl(P)$, is the complement of the interior of its complement with respect to X.

$$cl(P) = (int(P^c))^c \tag{2.3}$$

where

$$P^c = X - P \tag{2.4}$$

It can be proven that a set P is closed if and only if $P = cl(P)$.

Definition 8 The boundary of a set P is defined as

$$bd(P) = cl(P) - int(P) \tag{2.5}$$

The term frontier of a set is also used to denote the boundary of a set.

Definition 9 The boundary of an open n-ball centered at $\vec{p}$ is called the $n-1$-dimensional sphere of radius r centered at $\vec{p}$.

$$\vec{S}^{n-1}(\vec{p}, r) = \{\vec{q} \in \Re^n,\ ||\vec{p} - \vec{q}|| = r\} \tag{2.6}$$

In the sequel, whenever we refer to the n-ball we will mean the n-dimensional ball of radius 1 centered at $\vec{0} = (0, \ldots, 0)$.

$$\vec{B}^n = \{\vec{q} \in \Re^n,\ ||\vec{q}|| < 1\} \tag{2.7}$$

Similarly, the $n-1$-sphere is by convention the $n-1$-dimensional sphere of radius 1 centered at $\vec{0}$.

$$\vec{S}^{n-1} = \{\vec{q} \in \Re^n,\ ||\vec{q}|| = 1\} \tag{2.8}$$

Definition 10 A set P is called a bounded set if there exists a real number $r > 0$, such that $P \subset \vec{B}^n(\vec{0}, r)$.

Definition 11 If a set $P \subset \Re^n$ is both closed and bounded, P is called a *compact* set.

This definition of a compact set is actually a theorem, based on a more general definition of compactness [74].

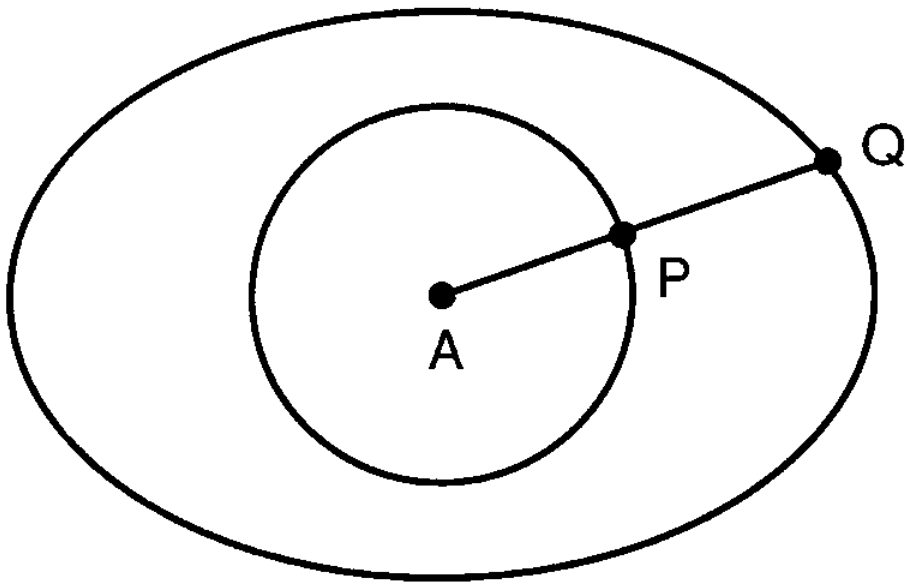

Figure 2.1: Homeomorphism between a circle and an ellipse

Definition 12 A point $\vec{x} \in X$ is called a limit or accumulation point of P if every neighborhood of $\vec{x}$ contains points of P other than $\vec{x}$. The closure of a set P contains P and all of its limit points.

Definition 13 A map $f : (X,T) \rightarrow (X',T')$ between two topological spaces (X,T) and (X',T') is continuous if for every $\vec{x} \in X$, every neighborhood of $f(\vec{x})$ has a neighborhood of $\vec{x}$ as a preimage. If f is bijective, i.e. one-to-one and the image of X under f is the entire X' and if f and its inverse f^{-1} are both continuous, then f is called a homeomorphism. A map f is continuous if for every open set $A \in T'$, $f^{-1}(A) \in T$. Two topological spaces are homeomorphic if there is a homeomorphism between them.

Definition 14 Two subsets P,Q of a topological space are called homeomorphic if there is a homeomorphism $h : X \rightarrow X$ such that $h(P) = Q$, i.e. the image of P under h is the set Q.

In view of the above, the one-dimensional sphere $\vec{S}^1$, i.e. the circle, is homeomorphic to an ellipse. The homeomorphism can be constructed as shown in Fig. 2.1. A finite cylinder or ellipsoid in $\Re^3$ are homeomorphic to a 3-ball, but a torus is not.

In the following, we introduce the concept of n-simplex. For this purpose, we consider a set X_n of $n+1$ points in space $\Re^n$, $X_n = \{\vec{x}_1, \ldots, \vec{x}_{n+1}\}$, $\vec{x}_i \in \Re^n$, $1 \leq i \leq n+1$, $\vec{x}_i = (x_{i_1}, \ldots x_{i_n})$, which do not lie in the same $n-1$ dimensional hyperplane. This is equivalent to

$$det[M] \neq 0 \tag{2.9}$$

where

$$[M] = [\vec{Q}_{n+1}\ \vec{Q}_n\ \ldots\ \vec{Q}_1\ \vec{1}] \tag{2.10}$$

$$[\vec{Q}_i] = [x_{1_i}\ \ldots\ x_{n_i}]^T \tag{2.11}$$

$$[\vec{1}] = [1 \ \ldots \ 1]^T \tag{2.12}$$

Definition 15 The n-simplex Δ_X^n is the set defined by

$$\Delta_X^n = \{\vec{x} = \sum_{i=1}^{n+1} \lambda_i \vec{x}_i, \ \lambda_i \in \Re^+, \ \sum_{i=1}^{n+1} \lambda_i = 1\} \tag{2.13}$$

Thus, the n-simplex is the convex hull of points $\vec{x}_i$, $1 \leq i \leq n+1$. Any two simplices Δ_X^n, Δ_Y^n, are homeomorphic to each other. A homeomorphism h between those simplices is defined by

$$h(\sum_{i=1}^{n+1} \lambda_i \vec{x}_i) = \sum_{i=1}^{n+1} \lambda_i \vec{y}_i \tag{2.14}$$

Thus, we may refer simply to the n-dimensional simplex Δ^n of $\Re^n$. Δ^n is homeomorphic to the closure of the n-ball. Further, its boundary $bd(\Delta^n)$ is homeomorphic to the $n-1$-sphere. The set $bd(\Delta^n)$ consists of all lower dimensional simplices. Each subset of $m+1$ points from $\{\vec{x}_1, \ldots, \vec{x}_n\}$, $m \leq n$, determines another m-dimensional simplex $\Delta_X^m \subseteq \Delta_X^n$.

Definition 16 If $X_m = \{\vec{x}_{i_j}, \ i_j \in \{1, \ldots n\}, \ 1 \leq j \leq m+1\}$, $X_m \subseteq X_n$ then the m-dimensional face Δ^m of Δ^n defined by X_m is the set obtained by setting $\lambda_k = 0$ in (2.13), where $k \neq i_j$, $1 \leq j \leq m+1$. This face is homeomorphic to an m-simplex Δ^m.

A simplex Δ^2 in $\Re^2$ is a triangular area ABC with non-collinear vertices A, B, C. Its one-dimensional faces are the linear segments AB, AC and BC and its zero-dimensional faces the vertices A, B, C. Similarly, a simplex Δ^3 in $\Re^3$ is the tetrahedron defined by four non-coplanar points A, B, C, D. Its two-dimensional faces are triangles, defined by sets containing any three of the above points, its one-dimensional faces are linear segments connecting any two of the vertices and the zero-dimensional faces are the vertices A, B, C, D.

Definition 17 An n-manifold $M \subseteq \Re^k$ with $k \geq n$ is a set, whose every point has an (open) neighborhood homeomorphic to the (open) ball $\vec{B}^n$. The boundary of manifold M, $\partial(M)$ is the set of those points $\vec{p}$ that have an (open) neighborhood homeomorphic to the half-n-ball, $\vec{HB}^n$ defined as

$$\vec{HB}^n = \{(x_1, \ldots, x_n) \in \Re^n, \ \sum_{i=1}^{n} x_i^2 < 1, \ x_1 \geq 0\} \tag{2.15}$$

The sets $\Re^n$, $\vec{S}^n$ are n-manifolds. The interior of a torus in $\Re^3$ is a three-manifold and the torus surface a two-manifold. The set $int(\Delta^n)$ is also an n-manifold.

Definition 18 An n-manifold with boundary is a set $\bar{M} \subseteq \Re^k$ with $k \geq n$, whose every point has a neighborhood homeomorphic either to the open ball $\vec{B}^n$ or to the half-n-ball $\vec{HB}^n$.

Figure 2.2 shows some examples of non-manifold point sets.

Definition 19 An n-dimensional simplicial complex C is a finite union of simplexes of dimension $\leq n$ such that all faces of any simplex of C are also in C and the intersection of any two simplexes of C is either the empty set or a face of each one of them. The point set defined by a simplicial complex C is denoted by $|C|$ and is called the *polytope* of C.

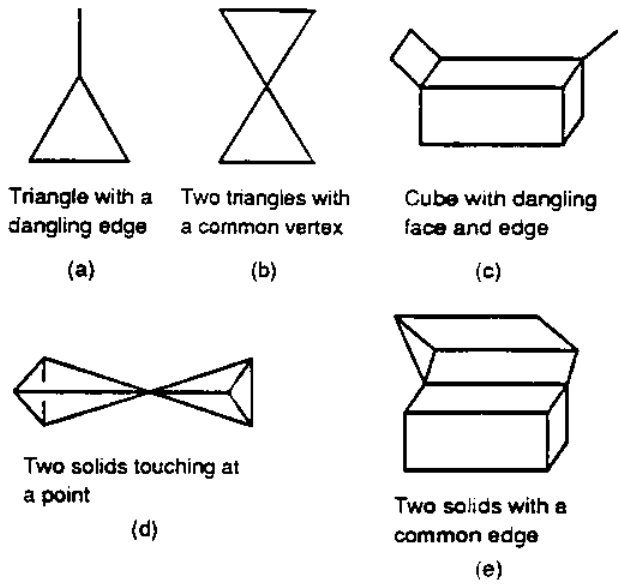

Figure 2.2: Non-Manifold point sets

The shapes of Figure 2.2 are all simplicial complexes. Rectangles, cubes, and prisms in particular (Figures 2.2c, 2.2e) may be easily decomposed intro simplexes of the appropriate dimension (triangles or tetrahedra). A sphere or a torus is not a simplicial complex as it cannot be decomposed into a finite number of simplexes. The set consisting of an n-simplex and all of its faces is a simplicial complex. Fig. 2.3 shows examples of point sets that are not simplicial complexes. In the left example, the intersection between the two triangles is not a face of either triangle. In the right example, the intersection between the triangle and the line segment is a face of the segment but not of the triangle.

Definition 19a An n-dimensional simplicial complex is regular if every one of its simplexes belongs to the boundary of at least one n-simplex. Thus, a regular simplicial complex is a collection of n-dimensional polyhedra and their boundaries and contains no isolated simplexes of a lower dimension.

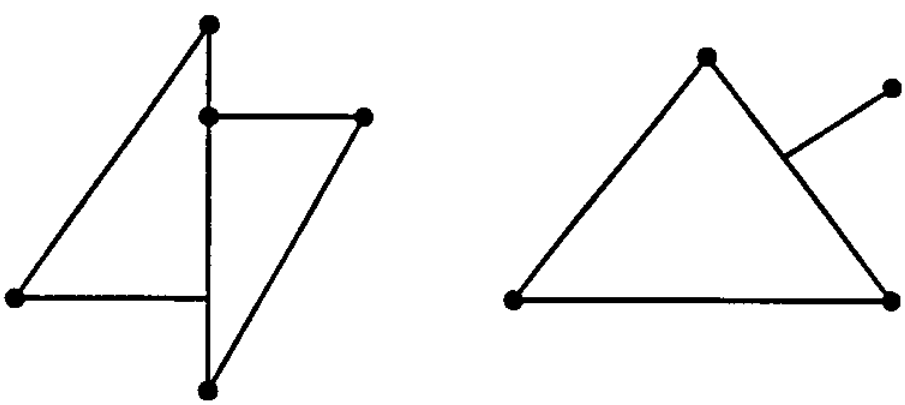

Figure 2.3: Point sets that are not simplicial complexes

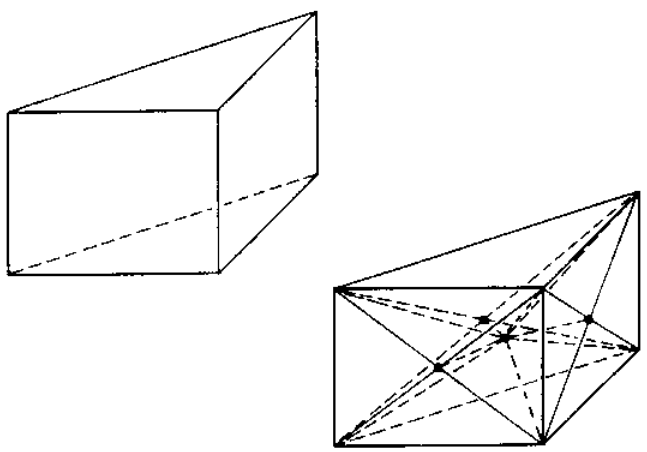

Figure 2.4: An example of triangulation of a triangular prism, adapted from Munkres, *Elements of Algebraic Topology*

Definition 20 A set $E \subseteq \Re^n$ is called triangulable if there exist a simplicial complex C and a homeomorphism $h : |C| \rightarrow E$. The simplicial complex C is called a triangulation of E. See Fig. 2.4 for an example.

It has been shown that any one, two, or three-manifold is triangulable [39]. Thus, triangulable simplicial complexes encompass a fairly large class of objects of interest in solid modeling.

An n-simplex Δ^n may be assigned two orientations. If we consider a permutation of its vertices $\vec{x}_{i_1}, \ldots, \vec{x}_{i_{n+1}}$ with $i_j \in \{1, \ldots, n+1\}$, $1 \leq j \leq n+1$ and pairwise different, we may assign the orientation sign according to the permutation index. It is recalled here, that a permutation is classified according to the number of interchanges of successive elements required to restore the original order, $\vec{x}_1, \ldots, \vec{x}_{n+1}$. If the number of element interchanges is even, the permutation is even, otherwise the permutation is odd. For example, in the case of a three element set, permutations $(\vec{x}_1, \vec{x}_2, \vec{x}_3)$, $(\vec{x}_3, \vec{x}_1, \vec{x}_2)$, $(\vec{x}_2, \vec{x}_3, \vec{x}_1)$, are even, whereas permutations $(\vec{x}_1, \vec{x}_3, \vec{x}_2)$, $(\vec{x}_2, \vec{x}_1, \vec{x}_3)$, $(\vec{x}_3, \vec{x}_2, \vec{x}_1)$ are odd. A positive orientation corresponds to an even permutation of the vertices and a negative orientation to an odd permutation. Fig. 2.5 depicts the two possible orientations of a triangle (two-simplex). An n-simplex Δ^n induces an orientation to any one of its faces Δ^j, $j \leq n$ by omitting the vertices not contained in Δ^j. The orientation of the triangle (Fig. 2.5) induces an orientation to its edges.

Definition 21 An n-dimensional simplicial complex is called orientable if all of its n-simplexes can be oriented such that opposite orientations are induced to any $(n-1)$-dimensional face shared

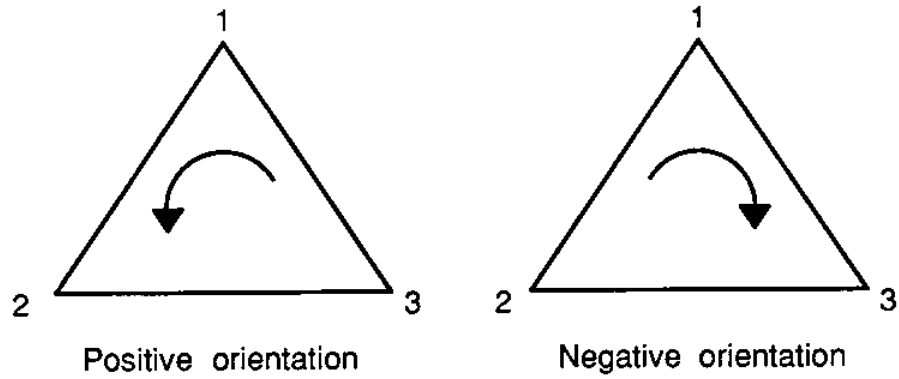

Figure 2.5: Orientation of a two-simplex

by two n-simplexes.

Since every n-manifold with $n \leq 3$ is triangulable, we may define orientability of such manifolds in terms of their triangulations. An n-manifold with $n \leq 3$ is orientable if one of its triangulations is orientable. Examples of non-orientable manifolds are the Möbius strip (manifold-with-boundary) and the Klein bottle [71, 75].

Definition 22 An open n-cell is a set homeomorphic to an open n-ball. A finite CW complex or simply n-complex is a pair (X, C), where X is a Hausdorff topological space and C a union of a finite number of disjoint open cells, of dimension lower or equal to n, $C = \{c_i,\ 1 \leq i \leq l\}$ with $\bigcup_{1 \leq i \leq l} c_i = X$, such that

1. for each open k-cell c_{i_k}, $k \leq n$, there exists a continuous function $f_i : \vec{B}^k \rightarrow X$ that maps $\vec{B}^k$ onto c_{i_k} and $\partial(\vec{B}^k)$ onto a finite union of open cells of C each of dimension less than k. If f_i are homeomorphisms and the boundary of each cell is equal to a union of a finite number of open cells of C, then (X, C) is called a regular CW complex.
2. a set Y is closed in X if $Y \cap c_i$ is closed in $cl(c_i)$ for each i.

Finite CW complexes provide a generalization of the concept of simplicial complexes. All point sets of Fig. 2.2 are CW complexes, whereas those of Fig. 2.3 are not.

An alternative definition, equivalent to the above definition, is the following:

Definition 22a An n-complex is a pair (X, C), where X is a Hausdorff topological space and C a finite collection of open, disjoint cells, such that

1. for each cell $c \in C$, $\partial(c)$ is a union of elements of C and
2. for every $c, d \in C$ such that $cl(c) \cap cl(d) \neq \emptyset$, then $cl(c) \cap cl(d)$ is a union of elements of C.

Definition 23 A subdivided n-manifold is a pair (M, C), where M is an n-manifold and (M, C) a finite, regular CW complex. The pair (M, C) is also called an n-complex or n-manifold subdivision or a subdivision of M.

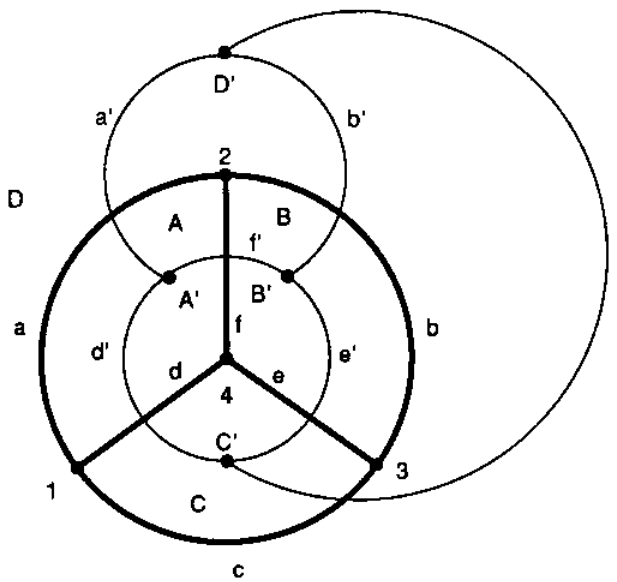

Figure 2.6: Subdivision of a disc and its dual

Definition 24 Two n-manifold subdivisions $(M, C), (N, D)$ are called equivalent if there is a homeomorphism between M and N carrying k-dimensional cells onto k-dimensional cells.

Definition 25 For every pair of cells $c, d \in C$ such that $c \subseteq cl(d)$, c is called a face of d. If, in addition, $c \neq d$, c is called a proper face of d.

Definition 26 Two cells are called incident if either of them is a proper face of the other. If $cl(c) \cap cl(d) \neq \emptyset$ and none of them is a proper face of the other, the cells c, d are called adjacent.

Definition 27 The dual of an n-complex (M, C) is an n-complex (M, C^*), for which a bijective mapping $f : C \rightarrow C^*$ exists such that

1. if c is a k-cell, then $c^* = f(c)$ is an $(n-k)$-cell and
2. if c, d are adjacent in C, then c^*, d^* are adjacent in C^*.

Fig. 2.6 shows a subdivision of the plane (two-manifold). Vertices (0-cells) are described by numbers, edges (1-cells) by small letters and 2-cells by capital letters. Cells A and a are incident, while cells A and B are adjacent. The dual subdivision is illustrated in the same picture. Cells dual to those of the primal subdivision are denoted by primed symbols. The dual subdivision is constructed in a similar manner to that of a dual to a planar graph and is explained later in this Section. The concept of dual manifold subdivision will be extended later to the dual complex.

Definition 28 A space X is connected if for any nonempty subsets A, B of X, such that $X = A \cup B$, then $cl(A) \cap B \neq \emptyset$ or $A \cap cl(B) \neq \emptyset$.

It can be proven that X is connected if and only if the only subsets that are both open and closed are X and the empty set. Another condition for connectedness of a space may be formulated through the path definition.

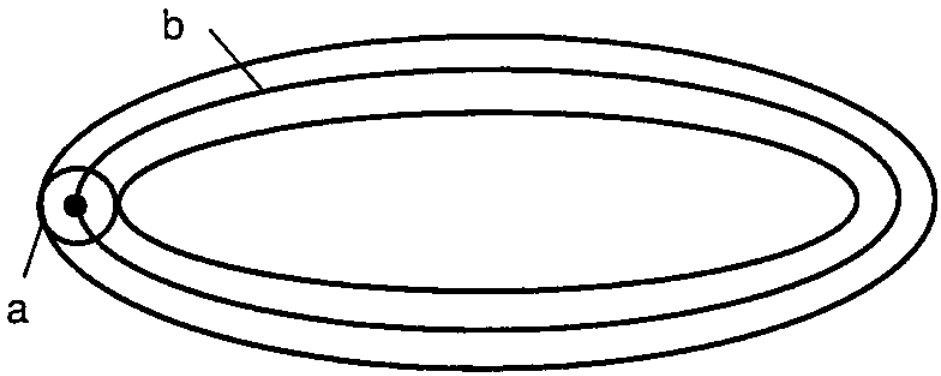

Figure 2.7: Genus of a torus

Definition 29 A path in X is a continuous function $\gamma : [0,1] \rightarrow X$. A space X is called path-connected if any two of its points can be joined by a path. If the space X is path-connected it can be proven that X is connected, but the converse is not always true [74].

Definition 30 Let C be an n-dimensional simplicial complex. Its Euler characteristic is defined as

$$\xi(C) = \sum_{i=0}^{n} (-1)^i \alpha_i \tag{2.16}$$

where α_i is the number of its i-dimensional simplexes. The Euler characteristic for surfaces, i.e. point sets homeomorphic to two-dimensional complexes is very important in solid modeling.

Any polyhedron or simplicial complex or 2-complex homeomorphic to a 2-sphere has an Euler characteristic equal to 2. Thus, by specializing (2.16) we obtain

$$V - E + F = 2 \tag{2.17}$$

where V, E, F are the number of vertices, edges and faces of the polyhedron, respectively.

Definition 30a The genus of a surface is the maximum number of closed curves that can be drawn on the surface, so that any two points on it can be connected with a path which does not cross any curve.

Thus, the genus of a sphere is 0, the genus of a torus is 1, since neither curve **a** or **b**, Fig. 2.7, divide the torus into two pieces. In general, an orientable surface is homeomorphic to a sphere with a finite number of handles. For example, a torus is homeomorhpic to a sphere with one handle.

Addition of a handle increases the genus of a surface by 1. The Euler characteristic of an orientable surface is $2(1 - G)$, where G is the genus of the surface. For a polyhedron homeomorphic to a sphere with G handles formula (2.17) becomes

$$V - E + F = 2(1 - G) \tag{2.18}$$

We can extend the Euler formula for polyhedra whose faces are bounded by more than one disconnected loops. A loop is a face boundary homeomorphic to a one- or zero-dimensional manifold. Further, we may also consider polyhedra with multiple shells, i.e. disconnected boundaries which are orientable manifolds themselves. If L is the number of loops and S the number of shells, the Euler formula becomes

$$V - E + F - H = 2(S - G) \tag{2.19}$$

where $H = L - F$. Euler operators transform a polyhedron while maintaining the validity of eq. (2.19).

A non-orientable configuration results if we cut a disk out of a sphere and replace it by a Möbius strip. The resulting surface is a closed manifold, which is called a projective plane or a sphere with a crosscap. The Euler characteristic of a sphere with n crosscaps is $2 - n$. Thus, for a polyhedron homeomorphic to a sphere with n- crosscaps, the Euler formula becomes

$$V - E + F = 2 - n \tag{2.20}$$

2.2 Graph Theoretical Concepts

Now, we are going to discuss a few graph theoretic concepts, [8, 45].

Definition 31 A simple graph or undirected graph is a pair (V, E), where V is a set of vertices and E a symmetrical, irreflexive relation in V, i.e. $E \subset V^2$, such that if $x \neq y \in V$ then $(x, x) \notin E$ and $(x, y) \in E \rightarrow (y, x) \in E$. All vertices x, y such that $(x, y) \in E$ are called adjacent.

The graph is usually represented as a set of points where adjacent vertices are connected by edges or arcs. An arc between vertices x and y represents both adjacent pairs (x, y) and (y, x). Fig. 2.8 is a simple graph with $V = \{1, 2, 3, 4, 5\}$ and $E = \{(1,2),(2,1),(1,5),(5,1),(2,3),(3,2),(2,4),(4,2)$ $(2,5),(5,2),(3,4),(4,3),(3,5),(5,3)\}$.

In a multigraph, two distinct vertices may be connected by more that one arc. A pseudograph is a multigraph with loops, i.e. arcs on the same vertex.

Definition 32 A directed graph or digraph is a pair (V, E), where V is a set and E an irreflexive relation on V. The arcs on a digraph are drawn with an arrow indicating the direction from the first to the second vertex in a pair $(x, y) \in E$.

Definition 33 Two graphs $(V, E), (V', E')$ are called isomorphic if there exists a bijective map $f : V \rightarrow V'$, which preserves adjacency.

Definition 34 A path of length k in a graph is a sequence of vertices $[v_1, v_2 \ldots v_k, v_{k+1}]$, such that $(v_i, v_{i+1}) \in E$, $1 \leq i \leq k$.

Definition 35 A simple path is a path where no arc is encountered twice.

Definition 36 A cycle is a simple path with coinciding end points.

Definition 37 A graph is connected if there is always at least one path connecting any two vertices.

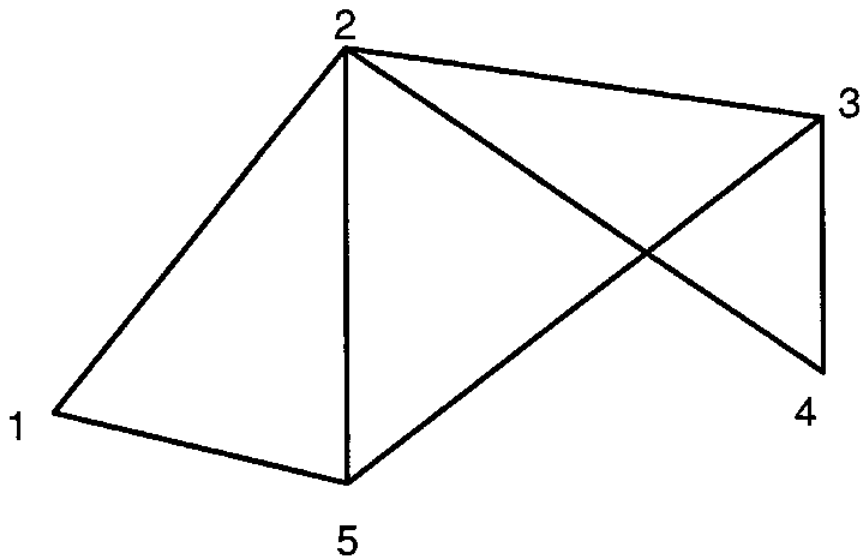

Figure 2.8: A simple graph

Definition 38 A tree is a connected graph with no cycles. If n is the number of its vertices, then the tree has exactly $n-1$ arcs.

Definition 39 A graph is planar if an arrangement of its vertices on a plane exists, such that its arcs can be drawn as curves not intersecting each other.

A planar graph partitions the plane into plane regions having arcs of the graph as boundaries. A polyhedron P which is homeomorphic to a sphere can always be represented by a planar graph (V, E). The vertices of this graph correspond to the vertices and its arcs to the edges of the polyhedron. The representation is established by bijective functions $f_V : V(P) \rightarrow V$, $f_E : E(P) \rightarrow E$, where $V(P), E(P)$ are sets containing the vertices and edges of the polyhedron, such that whenever $v_1, v_2 \in V(P)$ are connected by an edge e, then $f_E(e) = (f_V(v_1), f_V(v_2)) \in E(P)$. Fig. 2.9 shows the planar graph corresponding to an octahedron. The plane regions generated by the edges of the associated graph correspond to the faces of the polyhedron. Face f_6, the exterior region, corresponds to the non-bounded planar region adjacent to arcs e_6, e_7, e_9 and vertices v_4, v_5, v_6. Euler's formula in its simplest form, eq. (2.17), holds for a planar graph

$$v - e + f = 2 \tag{2.21}$$

where v is the number of vertices, e the number of arcs and f the number of planar regions.

A graph can be embedded on a surface, if it can be drawn on the surface such that no edges cross each other. A planar graph can always be embedded on a sphere. Any graph can be embedded on a sphere with handles, one handle for each crossing pair of edges when the graph is drawn on a plane. For any graph G there is an isomorphic graph G_0, which can be embedded on a sphere with a minimum number of handles. This number, $\gamma(G)$, is the genus of the graph and is equal to the genus of the surface on which it is embedded.

Any planar graph has an associated *dual* (pseudo)graph. The dual is constructed by placing

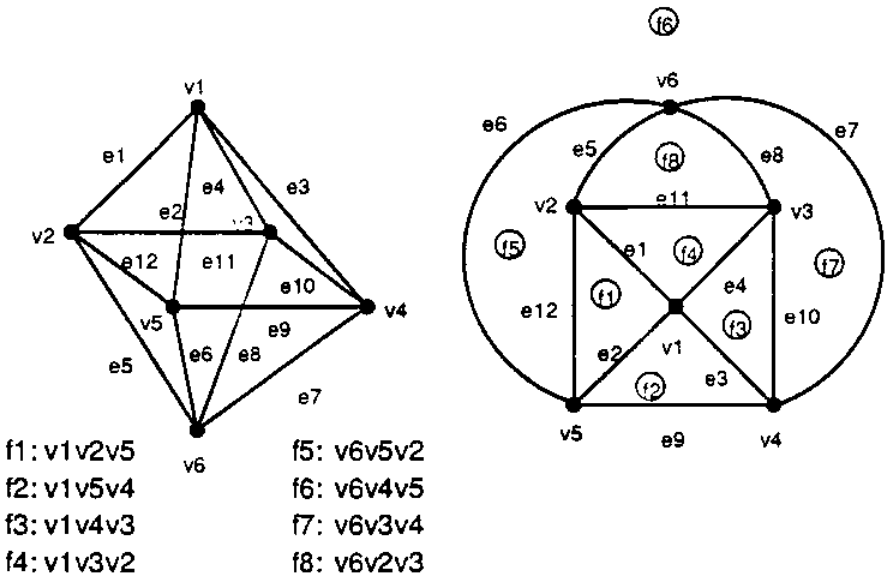

Figure 2.9: Octahedron and its planar graph

one vertex in each planar region including the exterior region, and connecting each pair of those vertices by one edge for every edge of the original graph at the boundary of the associated regions. Dual edges cross the associated edges of the original graph at one point only. Fig. 2.10a shows the graph of Fig. 2.9 and its dual. The vertices of the dual graph are unfilled circles and its edges are drawn with thin lines. Fig. 2.10b also shows a planar graph, whose dual is a pseudograph. The dual graph of a planar graph associated to a polyhedron P defines the dual polyhedron, $dual(P)$. The dual polyhedron is a simplicial complex homeomorphic to the spherical polyhedron resulting after embedding of the dual graph on the sphere surface. The dual to the graph of Fig. 2.9 corresponds to a cube, hence a cube is dual to an octahedron.

To any n-complex, a graph showing all incidence relations between k and $k-1$ cells, $0 \leq k \leq n$, is associated. The vertices of this graph are all the faces of the complex. An arc connects two incident k and $k-1$ dimensional faces. Usually, all vertices of the graph corresponding to k-dimensional faces are arranged on the same level. Fig. 2.11 shows the incidence graph of the octahedron of Fig. 2.9.

The concept of the dual polyhedron may be extended to complexes. Each k-cell of a complex is mapped to an $(n-k)$-cell of the dual. Incidence is preserved in the dual. The incidence graph of the dual is the "inverted" incidence graph of the primal complex.

We conclude this section with the definition of hypergraphs [8].

Definition 40 A hypergraph is a pair $V, \vec{E}$, where V is a set of vertices and $\vec{E} = \{E_i, i \in I\}$ a set of subsets of set V, $E_i \subseteq V$, $i \in I$, where I is an index set. The sets E_i are called hyperarcs. Most definitions and structures on graphs may be extended to hypergraphs.

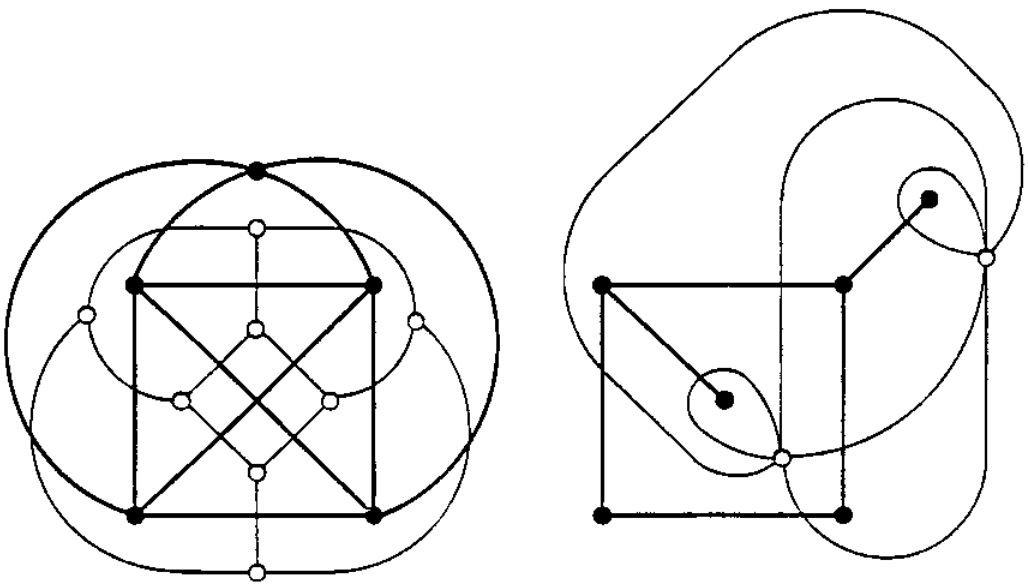

Figure 2.10: Planar graphs and their duals

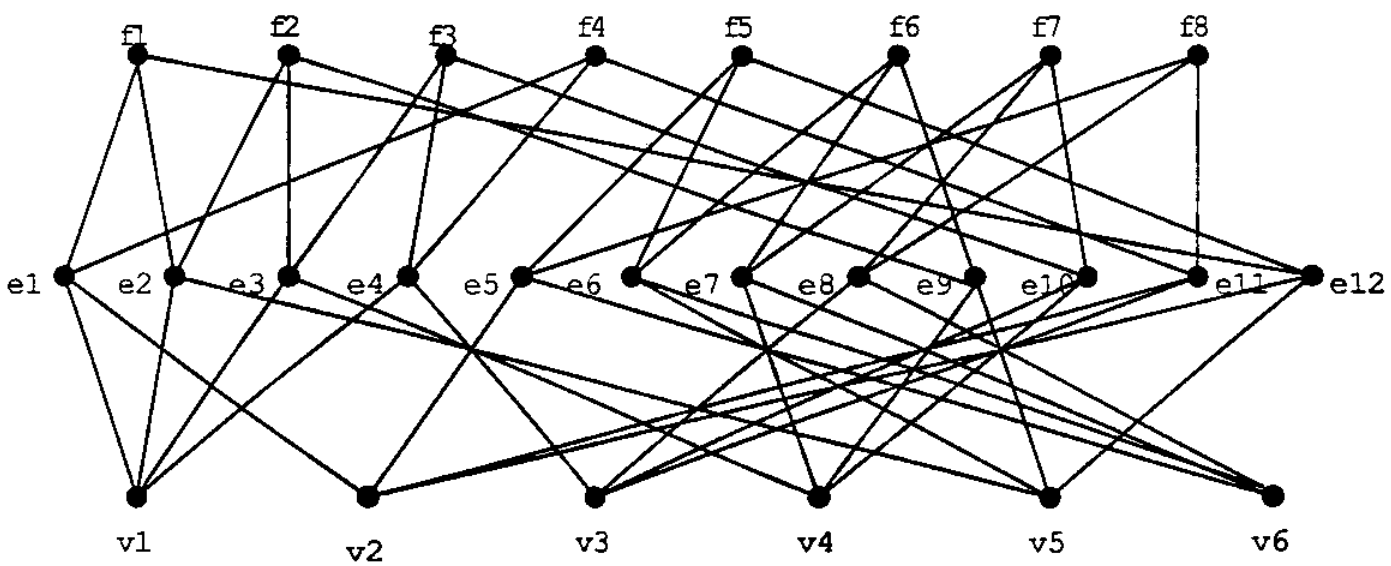

Figure 2.11: Incidence graph of an octahedron

Chapter 3

Boundary Representation Modeling Systems

In this section, we review some well known solid modeling systems. Each of them is characterized according to the class of solids (modeling space) it may handle. The data structure and special features of the system are described.

A distinction is made between Euler-based, non-manifold and abstract models. As mentioned in the introduction, Euler-based systems are built on Euler operators, which modify an object incrementally, such that the Euler formula in one of its forms (eqs (2.17), (2.18), (2.19)) is always valid. Non-Manifold models are capable of modeling non-manifold objects using special compound topological elements or the complete incidence graph. Abstract models use a single primitive structure and a limited set of constructors. The basic constructors and high level operators built on top of them modify the modeled object such that it always belongs in the topological class handled by the modeler.

3.1 Euler-Based Systems

3.1.1 The Winged-Edge Structure

One of the first models was the *winged-edge* structure by Baumgart, [7]. The modeling space consists of oriented, manifold polyhedra with planar faces. We assume that all faces are oriented consistently, as explained in the previous section. A diagram illustrating the winged-edge structure is shown in Fig. 3.1.

The center piece of the winged-edge structure is the edge, e, which is oriented. Letter "n" stands for next, "p" for previous, "cw" for clockwise, "ccw" for counterclockwise. Every edge is adjacent to two faces, *nface(e)* and *pface(e)*. Face *nface(e)* is the face, whose orientation induces on edge e an orientation coinciding with the original edge orientation, i.e. from *pvt(e)* to *nvt(e)*. Face *pface(e)* induces the opposite orientation to edge e. Edge *ncw(e)* is adjacent to *nface(e)* and *pvt(e)*, edge *nccw(e)* is adjacent to *nface(e)* and *nvt(e)*. Edges *pcw(e)*, *pccw(e)* are defined in an analogous manner.

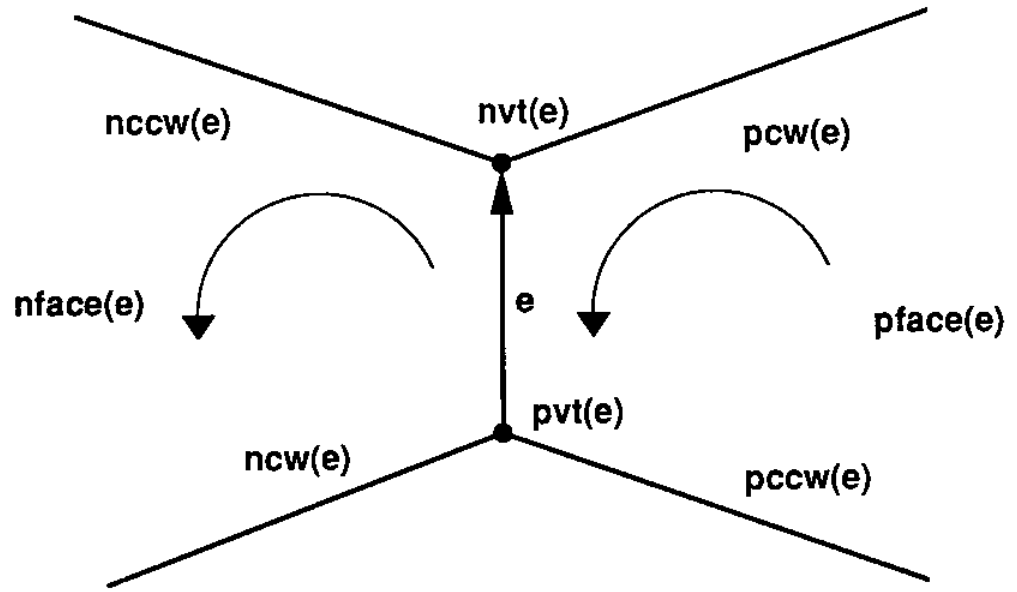

Figure 3.1: The winged-edge structure

The data structure comprises four classes of nodes. The body node points to three circular lists, the face, the edge and the vertex ring, containing all faces, edges and vertices of the polyhedron. This node contains also pointers which assign the body a place in a hierarchical assembly of bodies represented by a tree structure. The edge nodes contain pointers implementing the winged-edge structure, pointers to the previous and the next edge of the circular edge list and data useful for graphical display of the edge. The face node contains pointers to the next and the previous face in the circular face list, a pointer to one of its bounding edges and information used by graphical surface rendering algorithms. Finally, the vertex node contains pointers to the next and the previous vertex in the circular vertex list, a pointer to one of its incident edges and geometrical information.

The above data structure permits the reconstruction of all adjacency relations, i.e. constructing ordered sets of edges around a face, vertices around a face, faces incident to a vertex, edges incident to a vertex, edges adjacent to an edge, vertices and faces incident to an edge. Baumgart also provides Euler operators for the construction of the polyhedron while obeying the Euler formula, eq. (2.19) with the H term removed.

A system based on the winged edge structure is BUILD [11] developed at the Computer Laboratory of the University of Cambridge, UK, by Braid, Hillyard and Stroud. The winged edge data structure is extended there to incorporate loops enabling modeling of multiply- connected faces. Thus, *nface(e)* and *pface(e)* pointers in Fig. 3.1 point now to loops rather than faces. Loops point in turn to associated faces.

By the addition of loops, the modeling space includes some non-manifold conditions, such as two distinct loops touching at one point. The modeler is based on a set of low-level Euler operators maintaining the validity of eq. (2.19). A consistent naming convention for Euler operators is used, for example *mev* (make edge and vertex) creates a vertex and the edge joining the new vertex and an existing vertex, *mekh* (make an edge, kill hole loop) joins two vertices of two distinct loops by

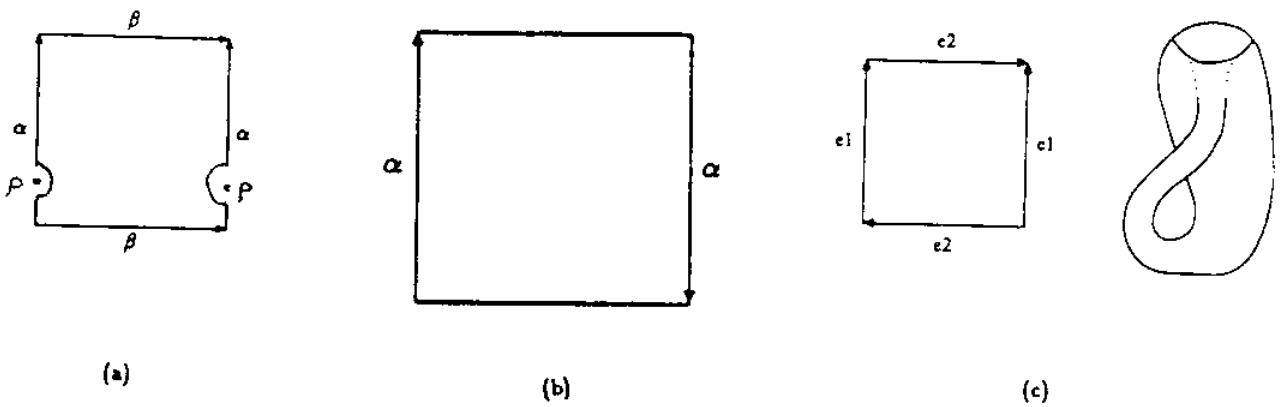

Figure 3.2: Plane models, adapted from Mäntylä

an edge.

Valid objects may be represented as a point on a hyperplane obeying eq (2.19) in a 6-dimensional space *(V,E,F,H,G,S)*. An Euler operator may be represented by a six-tuple, whose elements are the number of corresponding entities added to the object by the operator. For example, *mev* is represented by the tuple (1,1,0,0,0,0). The vector in $\Re^6$ normal to the hyperplane defined by eq. (2.19) is $v_0 = (1, -1, 1, -1, 2, -2)$. In [11] a set of five Euler operators is given, $eop_1, \ldots, eop_5$, which together with the above vector form a basis of the $\Re^6$ space. Therefore, any valid object may be expressed in terms of the new basis as $\lambda_1 eop_1 + \ldots + \lambda_5 eop_5 + \lambda_6 v_0$, where $\lambda_6 = 0$. Coefficients $\lambda_1, \ldots, \lambda_5$ are integers indicating the number of times the corresponding basic operator should be applied to construct the object.

High level operators include construction of planar shapes (lamina), sweeping operations for the construction of a solid from lamina, swinging operations for the construction of a solid of revolution by rotating lamina about a fixed axis, chamfering, i.e. replacing an edge or vertex by a face and tweaking, which is a generalized sweep operator. The modeler is capable of handling edges and faces with a non- planar geometry and provides high-level checking routines to detect invalid shapes such as self-intersecting faces. After successful termination of the checking routines, Boolean operations may be performed on solids. The original BUILD modeler evolved to the ROMULUS system.

Around the same time as Braid, Eastman, Weiler, Henrion, and Thornton [33, 35, 34] have extended Baumgart's structure working independently. Their system, GLIDE, may also handle loops and multiple shells.

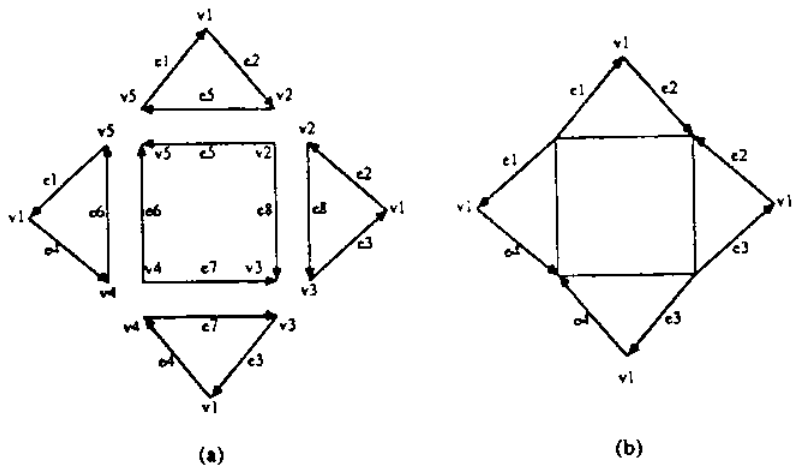

Figure 3.3: Plane model of a pyramid, adapted from Mäntylä

3.1.2 The Geometric Work Bench (GWB)

GWB [70, 69] is a polyhedral solid modeler developed by Mäntylä and Sulonen. The data structure is built around the plane model of a two-manifold, which is a graph-based approach to represent manifold structures, either orientable or non-orientable. In a plane model, each face is a directed, labeled graph, whose vertices are labeled as the vertices incident to the face and whose arcs are labeled as the edges incident to the face. Edges bearing the same label are topologically identified in a manner consistent with their orientation. Fig. 3.2a shows the plane model of a torus. The neighborhood of point P on edge α consists of the union of the two half disks around the instances of this point on the identified edges. To build the torus, the rectangle is wrapped such that edges α and β are brought in coincidence. Fig. 3.2b shows the plane model of a Möbius strip and Fig. 3.2c that of a Klein bottle. Fig. 3.3 shows the plane model of a pyramid.

A set of basic Euler operators for manipulating plane models are implemented in GWB. The data structure for the description of topological relations uses a hierarchical scheme with the following elements in decreasing order: solid, face, loop, half-edge, vertex. A solid node has pointers to a doubly linked list of solids coexisting in the same model and one pointer to each of three doubly connected lists of its faces, half-edges, and vertices. Each face node has one pointer to its parent solid, pointers to the previous and the next face in the face list, one pointer to a loop forming its outer boundary and one pointer to a doubly-linked list of loops consisting of all internal loops of the face. The face geometry is described by the four coefficients of the underlying plane equation. The loop node has one pointer to its parent face, one pointer to one of the half-edges belonging to its boundary and pointers to the next and the previous loop of the circular list containing the loops of the parent face. Each edge may be thought as split in two halves, each half associated with each loop the edge belongs to. Half-Edges forming a loop are arranged in a circular list. In the case of a face bounding loop, the arrangement of half-edges is compatible with the face orientation. A half-edge node contains a pointer to its parent loop, pointers to the previous and

next half-edge in the list, a pointer to its starting vertex in the direction of the loop and a pointer to its partner half-edge. Edges are also arranged in a doubly-linked circular list. Each edge node has pointers to the next and the previous edge in the list and one pointer for each half-edge. Finally, a vertex node has pointers to the next and previous vertex in the doubly-linked vertex list and a pointer to a half-edge having this vertex as a starting vertex. Further, a vertex node contains the homogeneous coordinates of the vertex. All adjacency relations may be reconstructed from the above structure.

High level operators in GWB include polygonal approximations of simple curved shapes, translational and rotational sweeping operators, and a gluing operator, i.e. merging of two solids over a common face, splitting a solid with a plane, slicing a portion of a solid and Boolean set operations on solids. A transaction log feature enables revoking the last operation (undo feature) as well as the implementation of an interrupt facility, which breaks an ongoing operation, such as a Boolean operation, and restores the model to the starting configuration.

3.1.3 The Hierarchical Face Adjacency Hypergraph (HFAH)

The Hierarchical Face Adjacency Hypergraph (HFAH) [2, 27] is another modeler for oriented two-manifold solids with curved, simply connected faces, developed by Ansaldi, De Floriani and Falcidieno. While all the previous models discussed so far are edge-based, HFAH is face-based. The model has two components, a hypergraph (FAH) and a tree representing a hierarchy of FAH's. The nodes (vertices) of the FAH are the faces of the polyhedron. For every edge, simple arcs connect face nodes having a common edge. All face nodes incident to a vertex are connected by a hyperarc. Arcs and hyperarcs incident to any face of the FAH are organized in an ordered sequence reflecting the ordering of edges and vertices in the loops of the face. Further, the nodes of a hyperarc are organized into an ordered sequence reflecting the ordering of faces around the vertex corresponding to the hyperarc. Further, arcs are labeled with the vertices of the edge and the loops containing the edge. If the face boundary consists of more than one loop, the arcs incident to the face are organized into distinct sequences, one for each loop.

The tree component of the HFAH models a hierarchy of FAH's, where each FAH represents a feature of the body, such as a depression, cavity, through-hole or protrusion. The root of the tree is a rough model of the body stripped of details of the above kind. Child nodes contain FAH's modeling features, which may in turn contain subfeatures, and so on. At the branches of the tree, parent-child relations are stored consisting of a mapping between associated vertices of the respective FAH's. This information is used to construct a new FAH of the body with the feature added. The procedure of inserting a feature in its parent node's FAH is called *refinement*. Details of the refinement algorithm are given in [27]. The inverse procedure, i.e. selecting a subgraph of a FAH, transforming the FAH and the subgraph and constructing the appropriate parent-child relationship, which is called an *abstraction* transformation, is also implemented.

Modeling a body in terms of a FAH representing its global shape and features by FAH's arranged in a hierarchical tree reflects the process followed by designers. The above model enables interrogating an object at various levels of abstraction. Unnecessary detail is hidden but may be recalled when needed. Low level Euler operators operating mainly on the faces of the object as well as high level operators, such as translational sweep, face glue and chamfering have been implemented [2].

3.1.4 Other Models

In [118], a data structure for two-manifold polyhedra having linear storage complexity with respect to the number of edges has been developed by Woo and Wolter. Further, the access time to find all adjacencies of one vertex, edge or face is linear on the average. The data structure contains two incidence relationships. The first relationship is $V \rightarrow E$, which associates to a vertex the list of edges adjacent to it and the second is $F \rightarrow E$, which associates to a face the list of its incident edges. In the structure, a vertex node contains the vertex coordinates and an ordered list of edge cells associated to its adjacent edges. Each edge cell contains one pointer to each incident vertex node, one pointer to each adjacent face node and one pointer to its geometry, a total of five pointers. The structure of a face node is analogous to that of a vertex node. In [118], it is shown that in a two-manifold polyhedron, the average number of edges adjacent to any vertex or face is 6. The linear storage complexity of the data structure follows from the above assertion and the fact that the numbers of faces, edges and vertices in a polyhedron are of the same order as implied by Euler's formula.

In [117], an analysis of the storage and time complexity of several boundary representation structures for two-manifold polyhedra is performed by Woo. It is shown there that all valid boundary representation structures have a linear storage complexity with respect to the number of edges of the polyhedron. The time complexity of algorithms for the reconstruction of all nine adjacency relationships between faces, edges and vertices varies between linear with respect to the number of edges and constant. One extreme of the scale is occupied by a data structure incorporating the $E \rightarrow V$ and $E \rightarrow F$ relationships, i.e. only the vertices and faces adjacent to each edge. Its storage complexity is $O(4E)$ and the time complexity to produce all nine adjacency relationships is $O(7E)$. On the other extreme a data structure including all nine adjacency relationships is found. Its storage complexity is $O(20E)$ and its time complexity is constant. The winged-edge structure has an $O(9E)$ storage complexity and an $O(6EV_i + 3k)$ time complexity, where EV_i is the number of edges adjacent to a vertex, 6 in the average, and k is a constant time unit for direct access of a node in the data structure. The maximum of EV_i is encountered in a pyramid with $E/2$ edges adjacent to its apex and the minimum in a tetrahedron with 3 edges per vertex. A better data structure in terms of storage and time complexity is proposed, the *symmetric data structure*, which in addition to the $E \rightarrow V$ and $E \rightarrow F$ adjacency relationships contains the $V \rightarrow E$ relationship, i.e. a list of edges adjacent to each vertex. Its storage complexity is $O(8E)$ and its time complexity $O(5EV_i + 4k)$. However, as it is pointed out in [117], the time complexity was calculated on the basis of the assumption that the frequencies of all query types are equal, which is not always true. Therefore, the application in hand should be analyzed first before deciding to use a specific data structure.

DESIGNBASE [23] is a solid modeler developed by Chiyokura, which uses a variant of the winged-edge structure. The modeling space is the same as in [11], i.e. two-manifold solids as far as topology is concerned. Curved geometry is supported. Faces do not appear explicitly in the data structure. They are represented by P-loops (parent-loops), a loop forming the face boundary and C-loops (child loops), internal loops of the face.

The data structure is relatively simple, consisting of four nodes. The solid node maintains a list of edges, loops and vertices. Many solid nodes may coexist in the same model. A loop node contains a pointer to one of its edges. If it is a P-loop, it contains a pointer to one of its C-loops.

If it is a C-loop, it contains a pointer to its parent P-loop and to the next C-loop in the C-loop list. An edge node implements a subset of the winged-edge structure. Only the *pccw(e), nccw(e), pvt(e)* and *nvt(e)* pointers appear (Fig. 3.1). The *nface(e)* and *pface(e)* pointers are replaced by pointers to the associated P-loops. The reduction of the original winged-edge structure speeds up updating operations but poses some restrictions on the modeling space. Thus, self-loops, i.e. edges with a single vertex are not allowed. Finally, the vertex node has a pointer to one of its adjacent edges.

In DESIGNBASE, a set of Euler operators has been implemented. High level operators include sweeps, rotations, gluing, mirroring, splitting and Boolean operators. Incorporation of curved edges and faces is done by special operators. Rounding and filleting have received special attention [22]. Gregory patches described by a control polyhedron are used as the curved surface primitive [24]. However, Boolean operators for curved surface configurations are not fully implemented yet. The user interface is CSG-like, i.e. it lets the user operate with volumes and produce local modifications by high-level operators.

Another solid modeler for objects bounded by free-form surfaces, Geomap-III, has been developed by Kimura [56]. The usual hierarchy of topological entities, ranging from solids to vertices is used in its internal object representation. Surfaces can be defined by curve meshes of an arbitrary topology. Operators for local shape modification (glue, drill, sweep, swing, round) have been implemented in Geomap-III. Shape definition by Boolean set operators is also possible.

Weiler [108, 112, 110, 109] was the first to present a data structure for the representation of non-manifold, three-dimensional objects. In his work, Weiler deals with both two-manifold polyhedra with curved surfaces and non-manifold objects. The non-manifold data structure is discussed in the next subsection. The manifold objects allow self-loops, more than one edge between two vertices and multiple shells. Thus, adjacency relationships of each shell can be represented by a pseudograph.

Several boundary representation structures for objects without loops and multiple shells but allowing self-loops and more than one edge between the same pair of vertices are investigated for topological sufficiency. Sufficient data structures involving one incidence relationship are the following:

- the $V < E >$ structure, consisting of the vertex-edge incidence relationship, i.e. a circular list of edges around each vertex. Faces are constructed by embedding the adjacency graph on a two-manifold surface.
- the $E\{< E >\}^2$ structure, where each edge is associated with two ordered lists containing the edges around each one of the two vertices of each edge.
- the $F < E >$ structure, consisting of the face-edge adjacency relationship, i.e. a cyclicly ordered list of edges around each face, consistent with face orientation.

All three above sufficient structures involve edges in their adjacency relationships. Pairs of the remaining adjacency relationships have also been investigated for sufficiency, but none was found sufficient. Further, Weiler investigated various edge-based data structures for two-manifold Boundary Representation. First, structures for polyhedra with self-loop edges and more than one edge between the same two vertices but not allowing loops and multiple shells are presented.

All data structures discussed below are edge-based, i.e. edges are the key for reconstructing all adjacency relationships and thus demonstrating topological sufficiency.

A common, support structure consists of a shell node pointing to one of its faces, a face node pointing to one of its bounding edges or a vertex if it consists of a single vertex. Faces are arranged in a circular list and each face node has, in addition, one pointer to the next and one pointer to the previous face in the list. Finally, there is a vertex pointer containing only the vertex coordinates and other data not related to topology. Geometrical information pertaining to faces and edges is stored at appropriate locations in the respective nodes.

Topological Sufficiency of the winged-edge structure [7] is demonstrated in [112] based on a theorem by Edmonds [36]. This structure is enhanced by adding new information to the $ncw(e)$, $nccw(e), pcw(e), pccw(e)$ fields (Fig. 3.1) indicating which side of the adjacent edge is pointed at. This improves efficiency of interrogation algorithms in curved geometry environments.

Another data structure is the vertex-edge structure. In this structure, each edge is represented by two nodes, each node corresponding to each one of its *edgeuses*. Each edgeuse is associated with each one of the vertices incident to the edge. Information stored in one edgeuse node comprises a pointer to the associated vertex and two pointers to the edges adjacent to the associated vertex as in the winged-edge structure, i.e. either $(nccw(e), pcw(e))$ or $(ncw(e), pccw(e))$, see Fig. 3.1. Further information stored in an edgeuse node includes a pointer to one of the adjacent faces and a pointer to the mate edgeuse. Sufficiency of this data structure is also demonstrated.

In the face-edge structure each edgeuse is associated with one adjacent face. This data structure is the dual of the previous one. Each edgeuse node contains one pointer for the previous and one pointer for the next edge around its adjacent face's bounding loop. More specifically, one edgeuse contains the pair $(pcw(e), pccw(e))$ and the other edgeuse the pair $(ncw(e), nccw(e))$, Fig. 3.1. The edgeuse points also to the other face than the one used to construct the previous pointer pair. Finally, there are pointers to one of the edge vertices and the mate edgeuse pointer. This data structure is also sufficient.

Edgeuses in the above data structures make interrogation algorithms more efficient and sufficiency proofs simpler. These data structures are extended to account for loops, multiple shells and regions. Loops may consist of an ordered edge list or a single vertex. Loops of the same face are arranged in a linear list. Shells are distinct three-dimensional solids bounded by a closed surface composed of faces, while regions are bounded or unbounded point sets of $\Re^3$, where each region may enclose one or several shells.

A loop node is added to the data structure, with a pointer to one of its edges or its single vertex, and another pointer to the next loop in the loop list. Further, the face node has a pointer to one of the loops in its loop list. In the previous data structures, all face pointers are replaced by appropriate loop pointers conforming to the hierarchical approach in B-rep, which requires the maintenance of incidence relationships between elements from higher to lower dimensions. Multiple shells can be arranged in a list or a tree implementing shell containment relationships [35] in a manner similar to the HFAH [2], described previously. A set of basic Euler operators maintaining the validity of eq. (2.19) has been implemented. In [35], the construction of higher level operators based on Euler operators is described.

Wilson [115] makes a detailed account of Euler formulas for wireframe objects which consist of vertices, edges and loops, and polyhedral objects. He also provides a complete set of Euler

operators for wireframe objects. In [114], an experimental neutral file format for transfer of models based on the B-rep representation is given. Topological information is encoded via incidence graphs having the basic entities (vertices, edges, loops, faces, shells) as nodes. Problems related to translation of topological structures from the neutral to a specific solid modeler's internal format and vice-versa are also discussed.

3.2 Data Structures for Non-Manifold Objects

During the last eight years some important contributions in data structures for non-manifold subdivisions have appeared. Examples of non-manifold configurations are dangling edges and faces, more than two faces meeting at the same edge, wire edges such as symmetry axes, faces touching at one point, etc. Non-Manifold modeling has broadened the scope of solid modeling to applications such as:

- coexistence of wireframe, surface and solid representation models;
- the same model can be used for finite element discretization and direct communication of the finite element analysis results to the original model;
- modeling of composite objects and representation of interior structures is incorporated in the same model; and,
- Boolean operators are closed in a non-manifold domain, therefore no need for regularization procedures exists.

There are some restrictions on the topology of the objects, which can be handled by the non-manifold models. Self-intersections of faces and edges are not allowed. Faces themselves should be bounded, manifold and homeomorphic to a planar shape. Thus, all non-manifold conditions occur at face boundaries and the genus may be inferred by topological information only. Shells are not allowed to intersect. Edges intersect faces only at their boundaries. In general, two topological elements may intersect each other along an element at least one level lower in the hierarchy than the lowest of these elements.

3.2.1 The Radial-Edge Structure

We continue here the review of Weiler's work with a description of the radial-edge structure. The major contribution of Weiler's thesis is a model for description of three-dimensional non-manifold topologies [112, 110, 109] using the *radial-edge structure*. There are certain restrictions on the topology of the objects, which can be handled by the radial-edge structure. Self-intersections of faces and edges are not allowed. Faces themselves should be finite, manifold and homeomorphic to a planar shape. Thus, all non-manifold conditions occur at face boundaries and the genus may be inferred by topological information only. Shells are not allowed to intersect. Edges intersect faces only at their boundaries. In general, two topological elements of the same dimension may intersect each other along an element of a lower dimension. If, however, two topological elements of different dimensions intersect each other, then they are either incident or they intersect in an

element of a lower dimension than the lowest of their dimensions. The hierarchy of topological elements is models, regions, shells, faces, loops, edges, and vertices. Additional elements that are used in the radial-edge structure are the faceuse, the loopuse, the edgeuse and the vertexuse. The faceuse represents the use of a face by a shell and carries an orientation consistent with the orientation of the shell containing the associated face on its boundary. Loopuses and edgeuses are likewise oriented consistently with the orientation of the parent faceuse or loopuse, respectively. An edgeuse may appear in a wireframe, where it is associated with each one of its vertices. A vertexuse is normally associated with an edgeuse. It may be associated with a loopuse or shell in degenerate loop and shell structures. Although the incorporation of uses in the radial-edge structure increases storage requirements, they facilitate traversals of the topological structure.

The radial-edge structure for a non-manifold subdivision consists of several circular lists, pointers from the top to the bottom in the hierarchy of elements and special arrangements of pointers in situations where several faces meet at an edge or several edges meets at a vertex. On top of the data structure is a pointer to one node of the circular model list. The model node points to the previous and the next model node of the circular model list and to one region of its circular region list. A region node points to the parent model node, the previous and the next region node in the circular region list and one shell of its circular shell list. The shell node points to its parent region node, the previous and the next shell node in the circular shell list, and to one node of the circular list composed of the shells' incident topological elements. These can be either faceuses, edgeuses or a single vertexuse. A face node points to one of its faceuse nodes and to a node, where geometrical information pertaining to the face is stored. The faceuse node has a pointer to the parent shell node, a pointer to the next and a pointer to the previous faceuse of a circular list of faceuses owned by the parent shell, a pointer to the mate faceuse, a pointer to one of its loopuses, a pointer to its parent face node and additional geometrical information. The absence of a pointer from a face node to a shell node should be noted. This occurs because a shell is bounded by faceuses, not faces. There are exactly two faceuses for each face, each one having an orientation opposite to the other. Each of two shells adjacent at a common face owns one of the faceuses. The loop node contains a pointer to one of its loopuses and geometrical information. The loopuse pointer contains a pointer to its parent faceuse node, a pointer to the mate loopuse node owned by the mate of the parent faceuse node, a pointer to the previous and the next loopuse of the circular loopuse list owned by the parent faceuse, a pointer to one of its edgeuses or to its vertexuse in case of a degenerate loop consisting of one vertex only and additional geometrical information. An edge node consists of a pointer to one of its edgeuses and geometrical data.

To visualize the structure of an edgeuse pointer we depict in Fig. 3.4 a non-manifold configuration, where several faces share a common edge. Faces are arranged in a radial ordering around the common edge. Each face is decomposed into two faceuses and each faceuse owns an instance of the same edge, i.e. an edgeuse. Edgeuses which are instances of the same edge are arranged in a circular list, where each edgeuse points to its mate edgeuse and the next edgeuse owned by an adjacent faceuse in the radial ordering. It is apparent that the number of edgeuses with the same parent edge is always even. Thus, an edgeuse node contains a pointer to an appropriate vertexuse in a manner consistent with the edgeuse orientation, a pointer to the mate edgeuse owned by the mate faceuse as shown in Fig. 3.4 or the other edgeuse in case of a wire edge, a pointer to the associated edge node and additional geometrical and orientation data. Further, if the edgeuse is

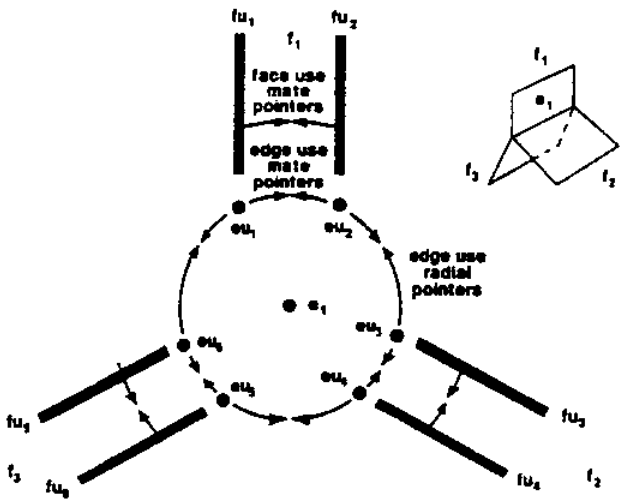

Figure 3.4: Faces sharing an edge in the radial-edge structure, adapted from Weiler

owned by a shell node in the degenerate case, the edgeuse node contains a pointer to it. In the normal case, there is a pointer to the owning loopuse, a pointer to the next and previous edgeuses of the circular edgeuse list belonging to the owning loopuse, and finally a pointer to the radially adjacent edgeuse (Fig. 3.4).

The vertex node contains a pointer to one of its vertexuses and geometrical information. In Fig. 3.5 a non-manifold situation is shown, where one vertex is owned by many edges in a wireframe. Vertexuses, i.e. instances of the same vertex are arranged in a circular list, where each vertexuse points to one of the adjacent edgeuses. In this case faceuses are absent, so the edgeuses are two instances of an edge, each associated with one of its vertices (vertexuses). The vertexuse contains a pointer to the previous and the next vertexuse node of the above circular list, a pointer to its parent vertex node, a pointer to either a parent shell or loopuse in degenerate cases, otherwise a pointer to its owning edgeuse, and additional geometrical information.

Geometrical information stored in an edge node may contain its parametric equation in the three-dimensional space, whereas geometrical information stored in one of its edgeuse nodes may be the equation of the edge in the owning face's parameter space. Similarly, geometrical information stored in a vertex node consists of its coordinates in a global system, whereas a vertexuse may contain the parameter value of the vertex in the parametric equation of its parent edgeuse.

In [112], sufficiency of the radial-edge structure is discussed. The issue of topological sufficiency of non-manifold data structures is still open. Further, traversal algorithms in the adjacency graph are given in [112]. A set of manifold and non-manifold Euler-like operators has been implemented in [112]. In [111], the construction of a generalized sweep operator is described based on the radial-edge data structure.

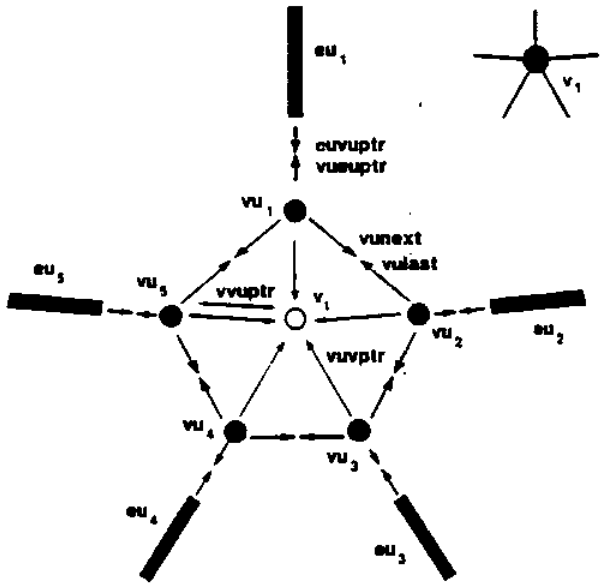

Figure 3.5: Edges sharing a vertex in the radial-edge structure, adapted from Weiler

3.2.2 The Tri-Cyclic Cusp Structure

Another data structure for non-manifold subdivisions is the *tri-cyclic-cusp* structure developed by Gursoz, Choi and Prinz [25, 43]. This structure is vertex-based and resolves ambiguities that may arise at a vertex with several adjacent faces touching at this vertex. The modeling space is composed of finite non-manifold solids with non-intersecting edges and faces as in [112]. Euler operators were not implemented in this model. Specific constructors and some Boolean operators are used instead.

The basic elements are vertex, edge, face, shell, region, wall, zone, disk, cusp, loop and edge-orientation. The hierarchy of these elements and relations between them in the tri-cyclic-cusp data structure are illustrated in Fig. 3.6. The first five of them have a meaning similar to the respective elements in the radial-edge structure. The wall is one of the two oriented sides of a face and is equivalent to the faceuse. An edge-orientation is one of the two possible orientations of an edge defined by an ordering of adjacent vertices. A zone is a three-dimensional local region around a vertex bounded by surfaces homeomorphic to conical surfaces with their apex at the vertex. Dangling faces and wire edges may belong to the boundary of a zone. A disk is the boundary of a zone. A conical surface adjacent to a vertex has two disks associated with it, one at the inside and one at the outside. Dangling faces and wire edges have only one disk. A cusp is the lowest element in the hierarchy, Fig. 3.6. It represents the use of a vertex and its adjacent edge in a loop bounding a wall. We may imagine a cusp as consisting of the intersection of a spherical neighborhood of the vertex with a wall of the face and the interior of the associated edge-orientation, i.e. the one not including the other vertex. The mate cusp involves the mate wall and the same vertex (Fig. 3.7). Isolated vertices and wire edges have no mate cusps. A disk adjacent to a vertex may thus be represented as a collection of cusps. A loop is a cyclic list of cusps with ordering consistent to the wall orientation. If the loop bounds a face, it has a mate loop, which is the border of the mate wall.

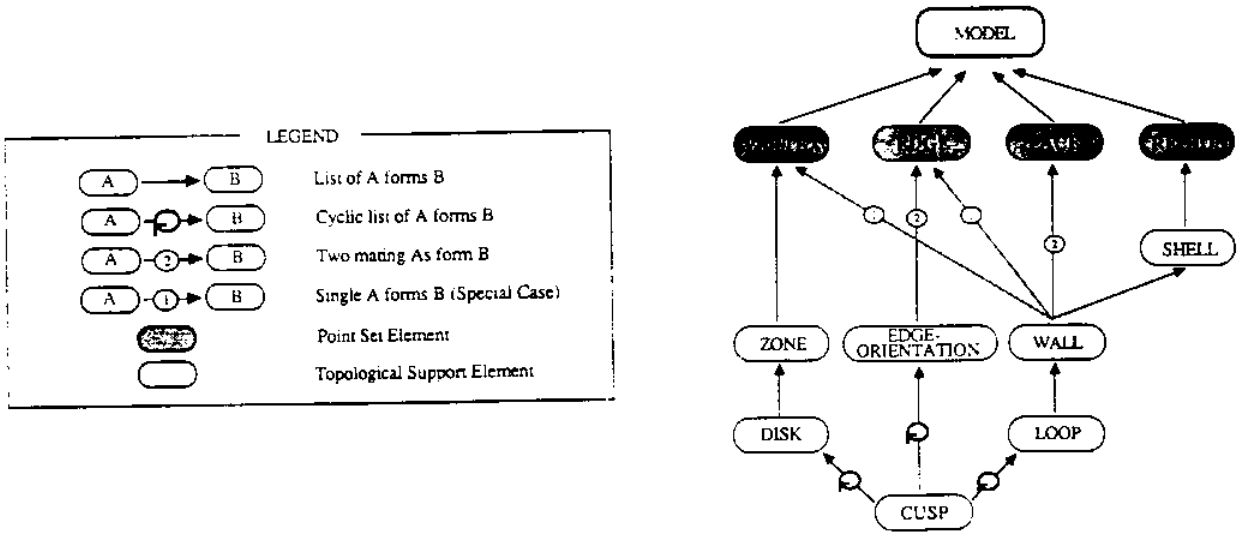

Figure 3.6: The tri-cyclic-cusp data structure, adapted from Gursoz

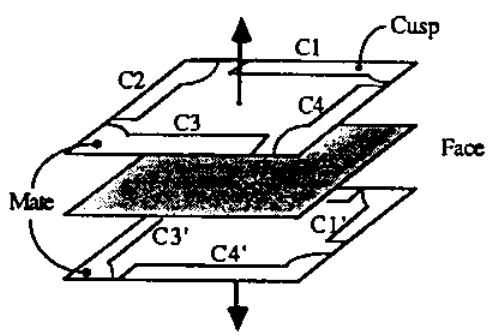

Figure 3.7: Cusp definition, adapted from Choi

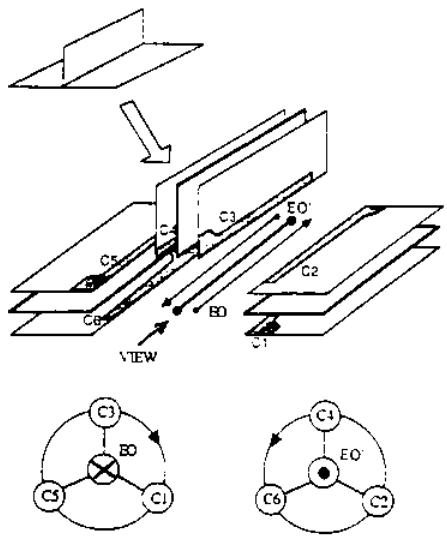

Figure 3.8: Edge-Orientation cycle, adapted from Gursoz

As it is implied by its name, the main element of tri-cyclic-cusp data structure is a set of three circular lists around each cusp. The disc-cycle is an ordered list of cusps forming a disc around a vertex. This cycle describes the arrangement of walls at the boundary of a zone around a vertex. The edge-orientation cycle is a circular list of cusps arranged around an edge orientation. Only cusps associated with the first vertex implied by the edge orientation are contained in the edge-orientation cycle. This cycle represents the topology in a situation similar to that of Fig. 3.4 in a different manner (Fig. 3.8). Two edge-orientation cycles are needed to describe local topology around the common edge. Traversal is made possible by navigating through each cycle and using the mate edge-orientation relationship. Finally, the loop cycle is a cyclic group of cusps around a loop consistent with the underlying wall orientation.

The remaining elements of the structure may be easily obtained from Fig. 3.6. A zone is a list of disks. One zone may contain more than one disk in a configuration, where one or more conical surfaces are contained in another. The neighborhood of a vertex is described by a list of zones. The two possible edge orientations point to the parent edge. The two walls of a face point to it in the general case, whereas a wire edge or an isolated vertex is adjacent to a single, unoriented wall. A shell is bounded by a list of walls and a region encloses a list of shells. Finally, a model is the root of the hierarchical structure and incorporates all elements of an object. Several models may coexist in the same modeling environment. Pointer associations not captured in Fig. 3.6 are between a cusp, an edge-orientation, a loop, a wall and their mates.

In [25], point set operators, such as the closure, complement, union, intersection, difference, disjoint and purge for non-manifold point sets are defined. The disjoint operator builds a partition of a group of objects to subgroups, where the elements of each subgroup are disjoint from elements of other subgroups. The purge operator purifies a group of elements by removing lower dimensional elements (such as an isolated vertex or a dangling edge on a face). In [44, 25] a general intersection algorithm for non-manifold boundary models in a linear geometry environment is described. NOODLES [42] is a modeler implemented on top of the tri-cyclic-cusp data structure.

In its original version, objects with linear edges and planar faces could be handled. Recently, it is being extended to incorporate curved geometry environments [19, 20].

3.2.3 Selective Geometric Complexes (SGC)

Rossignac and O'Connor [94] developed a model for describing non-manifold subdivisions in n dimensions. Objects called Selective Geometric Complexes (SGC) are used to model the subdivision. These are collections of open, connected cells exhibiting a structure similar to that of CW-complexes. Cells are allowed to have internal cracks represented by isolated cells of lower dimensions. All configurations discussed in the radial-edge data structure and the tri-cyclic-cusp structure may be modeled by SGC's. The data structure used for describing SGC's is graph based. Boolean operators and other transformations of non-manifold subdivisions may be expressed in terms of three basic operators, *subdivision, selection* and *simplification.*

To summarize the theory behind SGC's, some additional mathematical definitions are in order. A *real algebraic variety,* or simply *variety* in $\Re^n$ is the locus of the real roots of a finite number of polynomials in n variables with real coefficients. For example, a plane defined by the equation $a_1x + a_2y + a_3z + a_4 = 0$, a cylinder with equation $x^2 + y^2 - a^2 = 0$, and a cone given by $x^2 + y^2 - z^2 = 0$ are all varieties in $\Re^3$. The intersection curve between two cylinders of different radii expressed as algebraics is also a variety in $\Re^3$. A subset V of a variety W, which is also a variety is called a subvariety of W. If V is a proper subset of W, it is called a proper subvariety of W. A variety is called reducible if it may be expressed as a union of more than one proper subvarieties, otherwise it is an irreducible variety. All varieties given above, i.e. the plane, the cylinder and the cone are irreducible. The intersection between the cone and the plane $x = 0$ passing through its apex is a reducible variety. Its proper subvarieties are the straight lines $x = 0, y + z = 0$ and $x = 0, y - z = 0$.

Let S be the set of singular points of a variety, such as cusps, self-crossings and isolated lower dimensional pieces. Further, let $R = V - S$ be the set of regular points of a variety. R is a smooth manifold which can be decomposed in a finite number of open, connected components, the *extents* of the variety. The dimension of the variety is equal to the dimension of R. Similarly, S may be expressed as the union of connected subsets of extents of lower dimensional varieties. Thus, a manifold decomposition of variety V may be constructed. In general, a manifold decomposition of variety V is a finite set $\mathcal{M}$ of connected manifolds, such that for any manifold $M \in \mathcal{M}$, ∂M is a union of elements of $\mathcal{M}$. The plane is a variety of dimension 2 consisting of only one extent. The intersection of two cylinders is a variety of dimension 1 consisting of two extents, the two segments of the intersection. The manifold decomposition of the cone consists of three elements, the apex and the two segments of the conical surface.

A cell is a connected, open subset of an extent. The unique irreducible variety and the extent to which the cell belongs are denoted by *c.variety* and *c.extent*, respectively. A cell is allowed to enclose isolated cells of a lower dimension, which do not belong to its point set, such as a surface with isolated vertices or cracks. For example, consider the cone defined previously and the circle $x^2 + y^2 - 1 = 0$. The subset of the conical surface bounded by this circle and the apex is a cell according to the above definition. The circle and the apex do not belong to its point set. Thus, a cell of an n-dimensional extent may not be homeomorphic to the ball $\vec{B}^n$. This is the main difference between cells of a CW-complex and cells of a geometrical complex defined as follows:

A geometric complex or simply complex K is a finite collection of cells $(c_j)_{j\in J}$ such that

1. $\forall i, j \in J$ with $i \neq j$, $c_i \cap c_j = \emptyset$
2. the boundary of each cell c is a union of cells of K, i.e. $\partial c = \bigcup_{i\in J} c_i$
3. For the above cells $c_i \in \partial c$ either $c_i \subset c.extent$ or $c_i \cap c.extent = \emptyset$

The collections of cells c_i as in the above definition is denoted by *c.boundary*. Further, *c.star* is the collection of all cells of the complex containing c in their boundary. Thus, for any cells $b, c \in K$, $b \in c.boundary$ iff $c \in b.star$. Both operators, *boundary* and *star* define transitive relations in the geometric complex K. The dimension of a cell, *c.dimension*, is the dimension of *c.extent*. For example, let a be the apex of the cone, s the cell forming part of the conical surface as defined above, c the circular boundary of cell s and d the open disk bounded by c. The set $K = \{a, s, c, d\}$ is a geometric complex. The boundary of cell s is $s.boundary = \{a, c\}$. We may easily see that $c \in s.extent$ but $a \cap s.extent = \emptyset$. Similarly, $d.boundary = \{c\}$ and $c \in d.extent$. Further, $c.star = \{s, d\}$ and $a.star = \{s\}$.

Two complexes A, B are *equal* if they consist of the same collection of cells, i.e. $\forall a \in A, \exists b \in B$ such that $a = b$. Two complexes A, B are called *compatible* if $\forall a \in A, \forall b \in B, a \cap b \neq \emptyset \rightarrow a = b$. The lowest row of Fig. 3.11 shows two compatible complexes. A complex A is called a *refinement* of complex B if each cell of B is a union of cells of A. It is obvious, that the point sets of complexes A, B are equal. In view of the above definition, every complex is a refinement of itself. A complex A is called a *proper* refinement of complex B if A is a refinement of B and A is not equal to B.

A cell b is called a regular boundary of cell c if $c \in b.star$ and $c.dimension = b.dimension + 1$. A neighborhood relation may be defined for a pair of cells, where one of them is a regular boundary of the other, denoted by *b.neighborhood(c)*. This may have one of three values, *left, right* or *full*. The value *full* indicates a cell contained in the interior of the closure of its incident cell, such as a crack in the form of the curve located in the interior of a surface patch. If cell b is a subset of ∂c, the neighborhood relation may be defined in terms of the orientation of their varieties. For example, if a linear variety is a parametric curve, it is inherently oriented. A cell on this variety is a segment on the curve with no self-intersections and its boundary consists of its end points. The neighborhood of the vertex corresponding to the lowest parameter value has a neighborhood value equal to *left*, the vertex corresponding to the highest parameter value a neighborhood value equal to *right*. If this segment is a proper boundary of a face, the value of the neighborhood relation may be obtained by comparing the direction of the segment and the orientation of the face. For example, in the geometric complex of Fig. 3.9 *e6.neighborhood(f1) = left, e2.neighborhood(f1) = left, e4.neighborhood(f1) = right*. The neighborhood of a face at the boundary of a shell may be defined by examining whether the normal vector to the face, whose direction is consistent with the face orientation, points into the shell interior.

A Selective Geometric Complex, O, is composed of a complex, *O.complex* and a set of attributes attached to each cell of the complex. One important binary attribute is the *active* attribute, which if set FALSE signals that the cell should not be included in the SGC. Thus, the point set of an SGC is the union of the point sets of its cells, whose *active* attribute has a value equal to TRUE. For example, if c is a line segment and *c.active* = FALSE, c models a crack not included in the point set of the SGC.

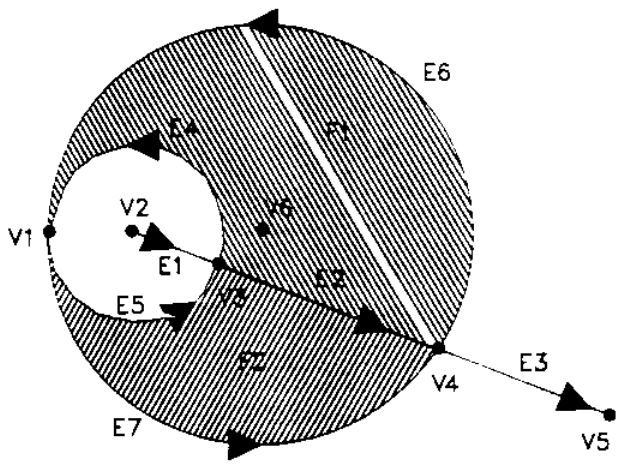

Figure 3.9: Two-dimensional geometric complex, adapted from Rossignac

The data structure for an unambiguous description of an SGC may implement one or more of three substructures,

1. boundary and star links between cells,
2. a set of references to all cells of the complex having the same dimension, and
3. a set of references to all cells belonging to the same variety or extents.

In [94], only the first substructure is considered. The data structure envisaged captures all boundary and star links with neighborhood information. Every cell c of the SGC is a node with a pointer to its extent and two lists, *bdry* and *star*, representing its boundary and star links, respectively. The *bdry* list is a list of unordered lists, where each of them contains boundary cells of the same dimension contained in *c.boundary*. Similarly, the *star* list is a set of unordered lists, each containing higher dimensional cells than c having the same dimension and belonging to *c.star*. The neighborhood relation is stored in the sublist containing cells of dimension *c.dimension+1*. The elements of this sublist are pairs consisting of a cell incident to c and the neighborhood relation between cell c and this cell. Additional information stored at the node is the dimension of the cell and its *active* and other attributes.

Fig. 3.9 shows a complex with 15 cells, i.e. 2 faces, 7 edges and 6 vertices. The pertaining nodes, and some of the information stored in them is as follows:

```
FACES
F1 : ext=plane, bdry=< <E2,E6,E4>,<V1,V3,V4,V6> >
F2 : ext=plane, bdry=< <E7,E2,E5>,<V1,V4,V3> >
EDGES
E1 : ext=line,  bdry=< <V2,V3> >
```

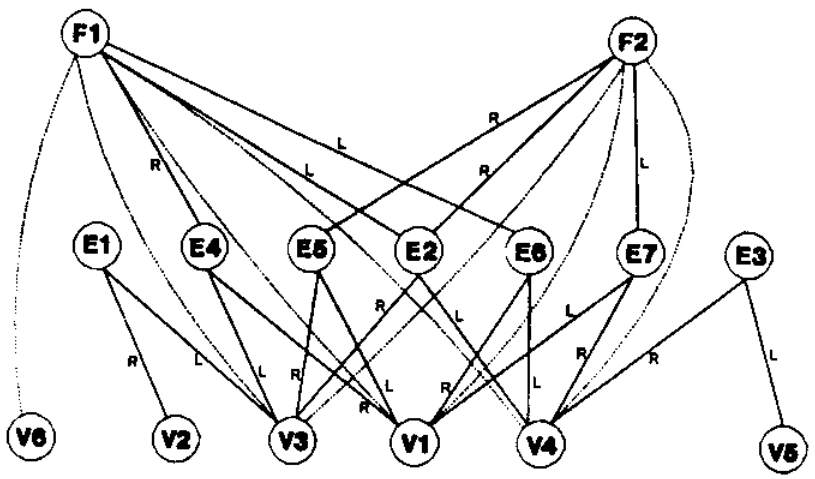

Figure 3.10: Adjacency graph of complex, adapted from Rossignac

```
E2 : ext=line,   bdry=< <V3,V4> >, star = < (F1,L),(F2,R) >
E3 : ext=line,   bdry=< <V4,V5> >
E4 : ext=cir1,   bdry=< <V3,V1> >, star = < (F1,R) >
E5 : ext=cir1,   bdry=< <V1,V3> >, star = < (F2,R) >
E6 : ext=cir2,   bdry=< <V4,V1> >, star = < (F1,L) >
E7 : ext=cir2,   bdry=< <V1,V4> >, star = < (F2,L) >
VERTICES
V1 : ext=pt1,    star=< <(E4,R),(E5,L),(E6,L),(E7,R)>,<F1,F2> >
V2 : ext=pt2,    star=< (E1,R) >
V3 : ext=pt3,    star=< <(E1,L),(E2,L),(E4,L),(E5,R)>,<F1,F2> >
V4 : ext=pt4,    star=< <(E2,R),(E3,L),(E6,R),(E7,L)>,<F1,F2> >
V5 : ext=pt5,    star=< (E3,R) >
V6 : ext=pt6,    star=< <F1> >
```

Fig. 3.10 shows the incidence graph of the complex, whose arcs connecting nodes of dimensions differing more than one are drawn with dotted lines. Solid arcs are labeled with the value of the neighborhood relation. It may be observed that some adjacency relationships captured in the data structure, such as those represented by the arcs (F2,V4), (F1,V4), (F1,V1), (F1,V3) are redundant, since they may be deduced by face-edge and edge-vertex incidence relations present in the graph. However, if the pair (F1,V6) were missing from the data structure, the complex would be ambiguous and geometrical information would be necessary to place vertex V6 correctly.

Three basic high level operators are defined in [94], *subdivision*, *selection* and *simplification*. These enable the implementation of any high level operator such as Boolean operators and interrogations, as any operator can be written in terms of the above three basic operators.

Subdivision takes as input parameters two geometric complexes A, B and produces refinements A', B' respectively, such that A', B' are compatible. The subdivision operator is implemented by

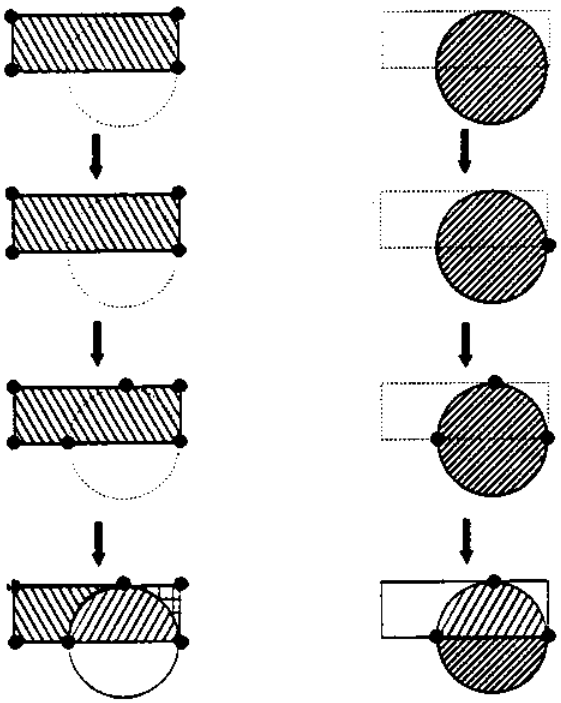

Figure 3.11: Refinement of a complex by subdivision, adapted from Rossignac

an algorithm that makes all cells of A of dimension k compatible with all cells of B of dimension $\leq k-1$ and vice-versa starting with $k=1$, i.e. subdivision of edges by vertices. Then, cells of dimension k in both A and B are subdivided at their common intersections. Fig. 3.11 shows the successive refinements of two two-dimensional complexes obtained by applying the above algorithm.

The *selection* operator merges several compatible complexes by deactivating common cells. Selection may be also applied to a single complex to sort out multiple occurrences of the same cell. The algorithm for establishing whether a cell a of a complex A is also a cell of complex B classifies one internal point of cell a against all cells of B. If this point is also an internal point of a cell b of B, then cells a, b must be identical since complexes A, B are compatible. All adjacency graphs are merged into one graph with multiple cells removed.

The *simplification* operator computes a new complex A' having the same point set as input complex A but as few cells as possible. This operator is implemented by application of three primitive operators, *Drop*, *Join* and *Incorporate*. The *Drop* operator removes all inactive cells of complex A. The *Join* operator removes an external boundary cell between two cells having the same extent provided that all three cells are active. The *Incorporate* operator removes an interior boundary cell by merging it to the unique cell that bounds it, provided that both cells are active. The output of the simplification operator is a simple cell, i.e. a cell for which no proper refinement may be constructed. Fig. 3.12 shows the resulting complex after successive application of the *subdivision*, *selection* and *simplification* operator.

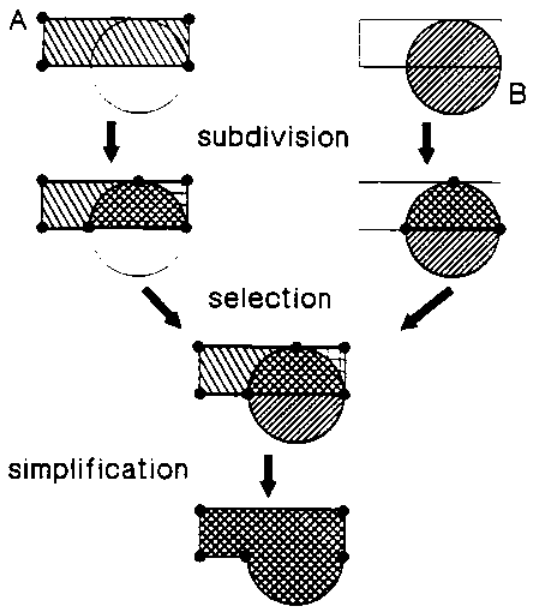

Figure 3.12: Merging of two complexes by primitive operations, adapted from Rossignac

In recent work, Wong and Sriram [116] developed an object-oriented product modeling framework, SHARED, which is designed to support cooperative product development. This system employs a non-manifold geometric modeler, GNOMES, described in He [46], and based on the SGC concept.

In [95], Rossignac and Requicha defined non-regularized set theoretic operators. The resulting sets are non-regular sets that may be described by SGC's. These operators may be implemented using the above three basic operators for SGC's.

3.2.4 Other Non-Manifold Models

In a recent work, Desaulniers and Stewart [28] have extended the Euler operators to r-sets [90]. Such sets are obtained by regularized set operations, mentioned in the Introduction in connection with the CSG representation. The set of manifold solids is a proper subset of r-sets. Configurations allowed here are solids touching at a vertex or along a common edge, like those depicted in Fig. 2.2. No isolated vertices or lines (such as symmetry axes) may be modeled. The data structure proposed in [28] is built on top of the GWB [69]. Additional elements are a list of manifold objects and a graph representing the spine of the r-set. The spine is composed of vertices and edges, where non-manifold conditions occur. There are cross links between the spine elements and incident manifold solids. The regularized extension of an Euler operator constructs an infinitesimal face around the edge or vertex of the spine and updates the data structure appropriately.

Masuda [72] developed a data structure for description of 3-complexes according to Definition 22a, where internal loops in 2-cells (faces) and cavities and holes in 3-cells (volumes) are allowed. Non-Manifold solids that can be represented with this data structure include 1, 2 and 3-complexes with the above extensions regarding 2- and 3-cells. In particular, configurations such as those of Figures 3.4 and 3.5 belong to the modeling space.

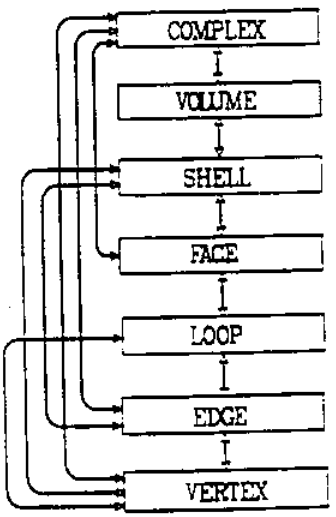

Figure 3.13: Hierarchical structure of topological elements, adapted from Masuda

The topological elements of the structure are, in hierarchical order, vertex, edge, loop, face, shell, volume and complex. Figure 3.13 shows the hierarchical structure of the topological elements. Every lower order element is related to the next higher element in the hierarchy, since it may belong to its boundary. A face is normally a boundary of one or two shells. A dangling face is related directly with the complex. A wire edge in the three-dimensional space or within a volume is connected with the associated degenerate complex or shell, respectively. In the normal case, it belongs to one or more loops. Finally an isolated vertex is either a degenerate loop within a face, a degenerate shell within a volume or an isolated point in the three-dimensional space.

Appropriate ordering and adjacency information is captured in the data structure. Ordering of edges in a loop and around a vertex as well as ordering of faces with a common edge using Weiler's radial edge structure [112] is maintained.

A set of extended Euler operators for incremental construction of complexes is defined. For this purpose, a modified Euler-Poincaré formula for non-manifold geometric models is used.

$$V - E + F - H - (v - v_h + v_c) = c - c_h + c_c \tag{3.1}$$

This is a specialization of the Euler-Poincaré formula for finite-cell complexes involving the i-dimensional Betti numbers b_i of the complex and the number of i-dimensional cells α_i [39]:

$$\sum_{i=0}^{n}(-1)^i\alpha_i = \sum_{i=0}^{n}(-1)^i b_i \tag{3.2}$$

In (3.1) V, E, F, and H have the same meaning as in equation (2.19). v, v_h, and v_c are the number of volumes, the number of holes through volumes and the number of cavities in volumes. c, c_h, and c_c are the number of complexes, the number of holes through complexes and the number of cavities in complexes, respectively.

Boolean operators for non-manifold objects are reduced to a merging and an extracting operation. Merging is performed by first generating all intersecting edges and vertices on each primitive object, unifying coinciding topological elements and reconstructing volumes. Pointers to the topological elements of the primitive objects are maintained for all elements resulting after the merging operation. Ordering information is produced and maintained as required.

Extracting operations are set theoretic operations between sets of topological elements resulting after the merging operations. Let A, B be the primitive objects and let $\Lambda(A), \Lambda(B)$ be the sets of topological elements belonging to A, B respectively in the merged object. Further, let $\Lambda(A^{in}) \subset \Lambda(A)$, $\Lambda(B^{in}) \subset \Lambda(A)$ be the topological elements that are not boundaries of other topological elements, such as dangling edges and faces or isolated vertices and volumes. The sets $cl[\Lambda(A^{in})]$, $cl[\Lambda(B^{in})]$ include all elements of $\Lambda(A^{in})$, $\Lambda(B^{in})$, and their boundary elements. Four Boolean operators are defined, union $\oplus$, difference $\ominus$ and two intersection operators, $\otimes^+$ and $\otimes^-$, where in the first of them wire edges and lamina faces are retained and in the second those are eliminated. If $+$, $-$ and $\cap$ denote the usual union, difference and intersection operations in sets, then the above Boolean operations are defined as

$$\Lambda(A \oplus B) = cl[\Lambda(A^{in}) + \Lambda(B^{in})] \tag{3.3}$$
$$\Lambda(A \ominus B) = cl[\Lambda(A^{in}) - \Lambda(B^{in})] \tag{3.4}$$
$$\Lambda(A \otimes^- B) = cl[\Lambda(A^{in}) \cap \Lambda(B^{in})] \tag{3.5}$$
$$\Lambda(A \otimes^+ B) = [\Lambda(A) \cap \Lambda(B)] \tag{3.6}$$
$$\tag{3.7}$$

Figure (3.14) shows an application of the above extracting operations. After merging, we have

$$\Lambda(A^{in}) = \{V_1, V_2, V_4, V_5, V_7, V_8\} \tag{3.8}$$
$$\Lambda(B^{in}) = \{V_2, V_5, V_8\} \tag{3.9}$$
$$\Lambda(C^{in}) = \{V_3, V_4, V_5, V_6, V_7, V_8\} \tag{3.10}$$
$$\Lambda(D^{in}) = \{V_3, V_4, V_5\} \tag{3.11}$$
$$\tag{3.12}$$

The final object R is defined as

$$R = C \ominus D \oplus A \ominus B \tag{3.13}$$

and thus

$$\Lambda(R^{in}) = \Lambda(C^{in}) - \Lambda(D^{in}) + \Lambda(A^{in}) - \Lambda(B^{in}) = \{V_1, V_4, V_6, V_7\} \tag{3.14}$$

The object R contains all topological elements of $\Lambda(R^{in})$ along with their boundaries.

The above definition of Boolean operators facilitates undoing operations, since information pertaining to the original primitive objects is maintained in the object resulting after the merging operation. This property is particularly useful in form-feature modeling.

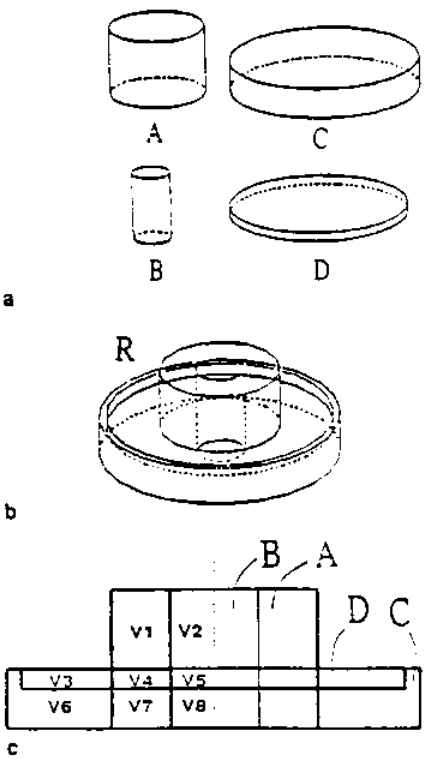

Figure 3.14: Resultant shape extracted from merged object: (a) four primitive objects (b) resultant object (c) volumes in merged object, adapted from Masuda

3.3 Abstract Models

We are now going to discuss abstract models for manifold subdivisions. The modeling space consists of two-manifolds [40], three-manifolds, [29, 62], and n-manifolds, [12, 13, 14, 65, 66]. Incremental construction of objects is made possible by using a small set of low level operators compared to the multitude of Euler operators necessary in Euler-based models. Most of these models were used to provide solutions to problems of computational geometry as construction of Delaunay triangulations and Voronoi diagrams.

3.3.1 The Quad-Edge Structure

The first model of the above kind, the forerunner of many models which appeared some years later, was the *quad-edge* structure, developed by Guibas and Stolfi [40]. The modeling space consists of subdivisions of compact, closed, two-manifolds (surfaces) either orientable or non-orientable, without boundary. Examples of such surfaces are the sphere, the torus and the projective plane. Subdivisions of surfaces of the above kind may be represented by the embedded graph corresponding to a polyhedron with edges homeomorphic to the open interval (0,1) and faces homeomorphic to an open disc. Both the primal and the dual graph are represented in the same model.

The basic constructing element is the edge, or, equivalently, arcs of the graph of the subdivision. Two directions may be defined on some edge, corresponding to each one of the two possible orderings of its vertices. Further, an adjacent face induces its orientation on the edge, therefore an edge may have two orientations. An orientation may always be defined locally on a manifold even if it is non-orientable. Intuitively, the orientation corresponds to the side of the surface one is looking at.

We refer to Fig. 3.15 for an overview of edge functions discussed below. Given a directed edge e we may characterize one vertex as its origin, *eOrg*, and the other as its destination, *eDest*. Together with the edge orientation, the faces on the left and the right, *eLeft*, *eRight* may be specified. The edge with the same orientation and opposite direction is *eSym*, that with the same direction and opposite orientation is *eFlip*. To visualize *eFlip* we may imagine that we look at the same edge from the other side of the surface without changing direction. A vertex may be oriented in a manner consistent with the local manifold orientation. Consider a neighborhood of vertex *eOrg*, which is homeomorphic to a disc, Fig. 3.16. The edges having a common orientation, which are directed away from the vertex form a circular list. The order of edges in this list is defined by traversing the boundary of the neighborhood in a counterclockwise direction, as shown in Fig. 3.16. The above circular list is the ring of edges out of a vertex. The edge immediately following e in this ring is *eOnext*.

If we traverse the left face, *eLeft*, in a counterclockwise direction, the first edge we encounter is *eLnext*. The direction and orientation of *eLnext* are such that *eLnextLeft=eLeft*. Edge *eDnext* is the next edge to e in the ring of edges around *eDest*. Edge *eRnext* is the next edge encountered by traversing *eRight* in a counterclockwise direction. By moving in a clockwise direction around a vertex or a face, we may define *eOprev*, *eDprev*, *eLprev* and *eRprev* in an analogous way.

The dual graph may define a dual subdivision, whose edges are associated one-to-one with the

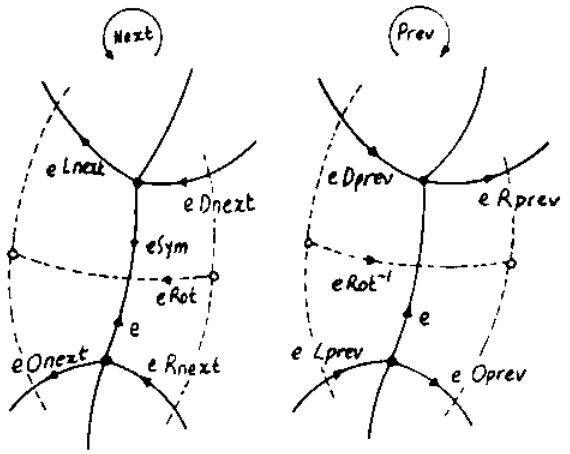

Figure 3.15: The edge functions, adapted from Guibas

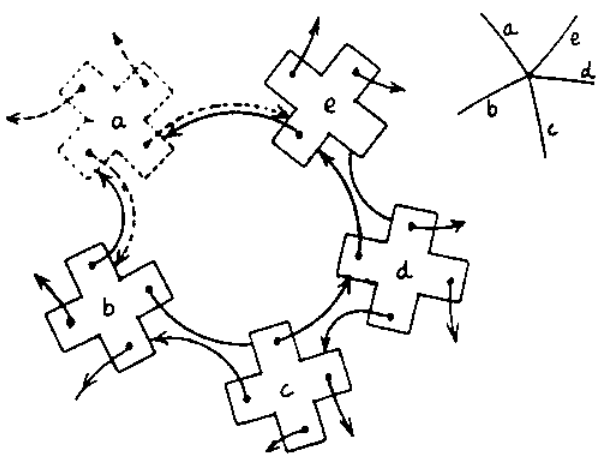

Figure 3.16: Ring of edges around a vertex, adapted from Guibas

edges of the primal graph by an appropriate construction explained in Section 2. The vertices of the dual subdivision correspond to faces of the primal one and vice-versa. The following relations define the dual subdivision S^* of a subdivision S on a two-manifold.

$$(eDual)Dual = eDual^2 = e \tag{3.15}$$

$$(eSym)Dual = (eDual)Sym \tag{3.16}$$

$$(eFlip)Dual = (eDual)FlipSym \tag{3.17}$$

$$(eLnext)Dual = (eDual)Onext^{-1} \tag{3.18}$$

Eq. (3.15) means that the dual of the dual subdivision is the primal subdivision. The *Dual* and *Sym* operators may be used interchangeably, eq.(3.16), but there is a direction reversion in the case of *FlipDual*, hence it is necessary to apply the *Sym* operator as in eq. (3.17). Eq. (3.18) implies that traversing the edges of a face in the primal graph in the counterclockwise direction is the same as moving around the corresponding vertex of the dual graph in the clockwise direction. Thus, the dual graph is embedded on the flipped manifold. For example, if we have a subdivision on the outside of a sphere, the dual subdivision is on the inside of the sphere. To avoid this flipping a new edge function, *Rot* is defined as follows:

$$eRot = eFlipDual = eDualFlipSym \tag{3.19}$$

Edge *eRot* is the "rotated" version of e and corresponds to an edge with same orientation as e which is directed from the dual vertex inside *eRight* to the dual vertex inside *eLeft* and crosses e at a right angle (Fig. 3.15). Thus, if *Rot* is applied to *eRot*, this is rotated counterclockwise and *eSym* is produced.

$$eRot^2 = eSym \tag{3.20}$$

We denote by $E(S)$ the set of edges of a subdivision. The edge functions satisfy the following properties:

$$eRot^4 = e \tag{3.21}$$

$$eRotOnextRotOnext = e \tag{3.22}$$

$$eRot^2 \neq e \tag{3.23}$$

$$e \in E(S) \Leftrightarrow eRot \in E(S^*) \tag{3.24}$$

$$e \in E(S) \Leftrightarrow eOnext \in E(S) \tag{3.25}$$

$$eFlip^2 = e \tag{3.26}$$

$$eFlipOnextFlipOnext = e \tag{3.27}$$

$$eFlipOnext^n \neq e \ \forall \text{ integer } n \tag{3.28}$$

$$eFlipRotFlipRot = e \tag{3.29}$$

$$e \in E(S) \Leftrightarrow eFlip \in E(S) \tag{3.30}$$

In [40] an abstract, combinational object is defined, the *edge algebra*. An edge algebra is a quintuple $(E, E^*, Onext, Rot, Flip)$, where E, E^* are finite sets and *Onext*, *Rot*, *Flip* are functions on E, E^* satisfying eqs. (3.21)-(3.30). It is proven in [40] that given an edge algebra, there is always a subdivision S on a two-manifold and a bijective mapping $f : E \rightarrow E(S)$ such that $f(eDual) = f(e)Dual \; \forall \, e \in E$. The proof is based on triangulations of subdivisions. Conversely, a subdivision S on a two-manifold defines an edge algebra in the natural way. Thus, an equivalence between manifold subdivisions and edge algebras is established.

All edge functions may be defined in an abstract way in terms of the three basic functions, *Onext*, *Rot* and *Flip*. The origin of an edge $e \in E$ is defined as the orbit of e under *Onext*, i.e. the sequence

$$eOrg = (e, eOnext, eOnext^2, \ldots, eOnext^{-1}, e) \tag{3.31}$$

The above list is finite, since E is a finite set. The remaining functions may be defined as follows

$$eSym = eRot^2 \tag{3.32}$$

$$eLeft = eRot^{-1}Org \tag{3.33}$$

$$eRight = eRotOrg \tag{3.34}$$

$$eDest = eSymOrg \tag{3.35}$$

$$eLnext = eRot^{-1}OnextRot \tag{3.36}$$

$$eRnext = eRotOnextRot^{-1} \tag{3.37}$$

$$eDnext = eSymOnextSym \tag{3.38}$$

$$eOprev = eOnext^{-1} = eRotOnextRot \tag{3.39}$$

$$eLprev = eLnext^{-1} = eOnextSym \tag{3.40}$$

$$eRprev = eRnext^{-1} = eSymOnext \tag{3.41}$$

$$eDprev = eDnext^{-1} = eRot^{-1}OnextRot^{-1} \tag{3.42}$$

The data structure for representing a subdivision and its dual is the *quad-edge* data structure. An array of dimension 4 is assigned to each edge. For an edge e, elements $e[0]$ through $e[3]$ contain data pertaining to e, $eRot$, $eRot^2 = eSym$ and $eRot^3 = eRot^{-1}$ respectively. The elements of the array are structures with two fields each, *Data* and *Next*. In field *Data*, geometrical and other non- topological information is stored. For example, $e[0]$ may contain pointers to the coordinates of the vertices of e, $e[1]$ a pointer to the curve geometry, $e[2]$ a pointer to the left face and $e[3]$ a pointer to the right face geometry specification. Field *Next* contains a pointer to $eOnext$. A bit f is associated with each array element indicating whether the flipped version of the edge should be taken. In view of the above, we may represent the basic structural element by a triple (e, r, f) corresponding to $eRot^r Flip^f$. All edge functions may be generated by the information stored in the data structure, for example

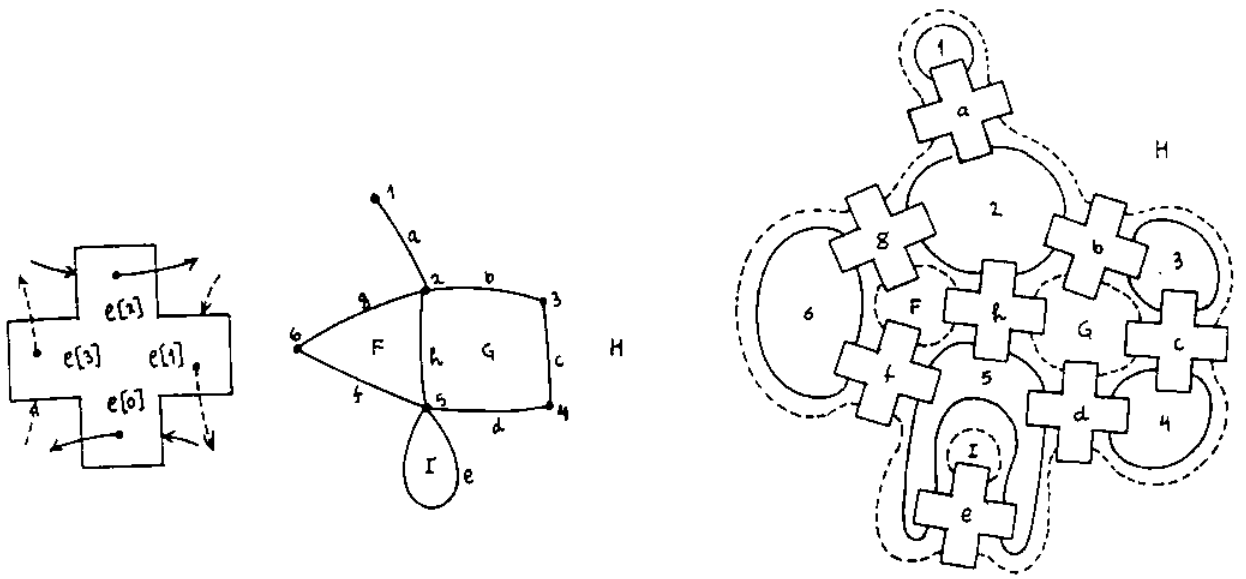

Figure 3.17: The quad-edge data structure, adapted from Guibas

$$(e,r,f)Rot = (e, r+1+2f, f) \tag{3.43}$$

$$(e,r,f)Flip = (e, r, f+1) \tag{3.44}$$

$$(e,r,f)Onext = (e_1, f, f) \text{ with } e_1 = e[r+f].Next \tag{3.45}$$

$$(e,r,f)Sym = (e, r+2, f) \tag{3.46}$$

$$(e,r,f)Rot^{-1} = (e, r+3+2f, f) \tag{3.47}$$

$$(e,r,f)Oprev = (e_1, 1-f, f) \text{ with } e_1 = e[r+1-f].Next \tag{3.48}$$

The second element of each triple in the above relations is evaluated mod 4, the third mod 2. Eqs. (3.43-3.48) may be proven using the basic properties of the edge algebra functions, eqs. (3.21-3.30) and the definitions of the remaining functions eqs. (3.32-3.42). The data structure is illustrated in Fig. 3.17.

The four edges obtained after repeated application of *Rot* to an edge e are arranged around a cross. They are the nodes of a hypergraph. The solid hyperarcs of this hypergraph correspond to the vertices of the primal graph (faces of the dual graph) and the dotted hyperarcs to the faces of the primal graph (vertices of the dual).

For orientable manifolds, such as a sphere with or without handles, the data structure may be simplified by omitting the flip bit. In fact, two-manifold polyhedral objects we are interested in solid modeling are orientable, and therefore the simplified version of the quad-edge data structure applies to them. Thus, in the simplified version of the quad-edge structure (orientable two-manifold) only four nodes are required, e, $e1 = eRot$, $e2 = eRot^2$, $e3 = eRot^3$ and the *Next* field contains the values *e.Next=e, e1.Next=e3, e2.Next=e2, e3.Next=e1.* Yet another simplification results if only $eRot$ and $eRot^2$ are represented, but this will increase the time complexity of some traversal algorithms. The storage complexity of the quad-edge structure compares favorably with

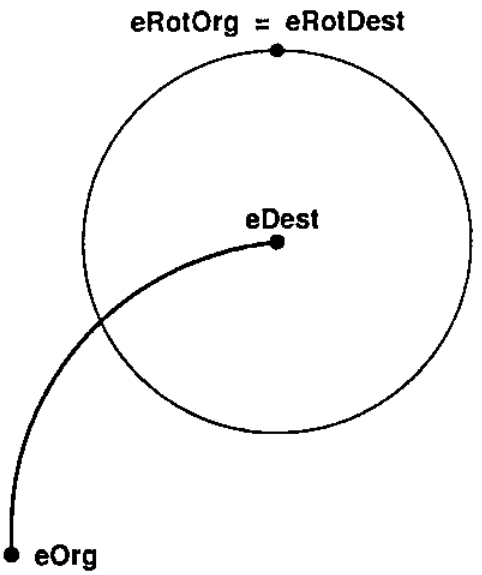

Figure 3.18: The *Makedge* operator

that of the winged-edge structure. While both require a linear storage space with respect to the number of edges N of the subdivision, the constant factor is smaller in the quad-edge structure.

Only two operators are needed to construct any subdivision. The first operator, *Makedge*, delivers a single edge, e, embedded on a two-manifold and its dual, *eRot*. The Dual is a loop, Fig. 3.18. The following relations hold for the edge delivered by *Makedge*

$$eLeft = eRight \tag{3.49}$$

$$eOnext = e \tag{3.50}$$

$$eRotOrg = eRotDest \tag{3.51}$$

$$eRotOnext = eRotSym = eRot^3 = eRot^{-1} \tag{3.52}$$

$$eRot^{-1}Onext = eRot^{-1}Sym = eRot \tag{3.53}$$

$$eSymOnext = eSym \tag{3.54}$$

The second operator is $Splice(a,b)$, which operates on the vertex and face rings of edges a, b, i.e. the lists $(a, aOnext, ..., aOnext^{-1}, a)$ $(aRot, aRotOnext, aRotOnext^2, ..., a)$ and the respective lists of b. The face ring of an edge is the vertex ring of its flipped dual, *aRot*. If the two rings are distinct, *Splice* combines them into one and if these are the same ring it breaks it into two pieces. If both rings are different, it may produce shapes that are of no interest in B-rep, such as crosscaps. The formal definition of *Splice* is as follows. If $\alpha = aOnextRot$ and $\beta = bOnextRot$, then $Splice(a,b)$ is the edge algebra with pairs $(aOnext, bOnext)$ and $(\alpha Onext, \beta Onext)$ swapped. In the general version of the quad-edge structure, i.e. with flip included, the following operations are carried out in addition

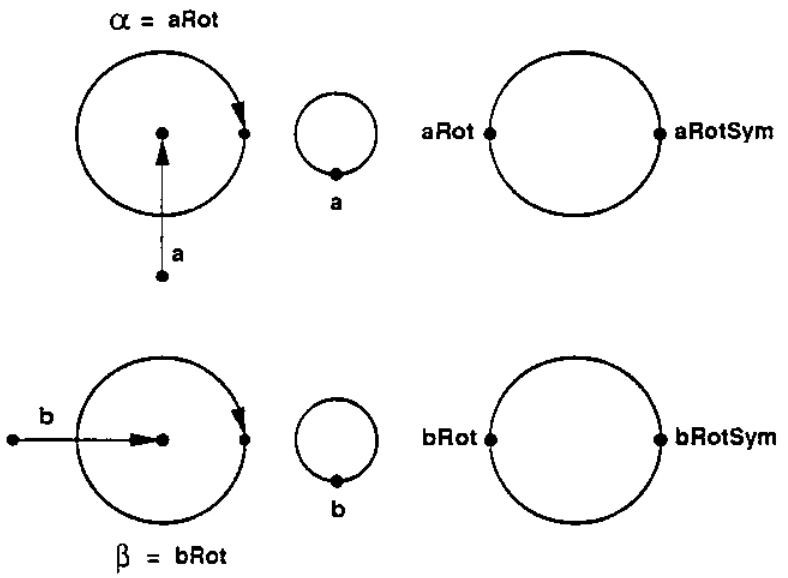

Figure 3.19: Joining of two edges

$$(bOnextFlip)Onext \leftarrow aFlip \tag{3.55}$$

$$(aOnextFlip)Onext \leftarrow bFlip \tag{3.56}$$

$$(\beta OnextFlip)Onext \leftarrow \alpha Flip \tag{3.57}$$

$$(\alpha OnextFlip)Onext \leftarrow \beta Flip \tag{3.58}$$

where a, b, α, β on the right hand side are the edges of the original algebra in all the above equations.

We illustrate the effect of *Splice* with some simple examples. In Fig. 3.19 two edges, a, b. are shown with their vertex and face rings. $Splice(a, b)$ unites the vertex and face rings pairwise. The result is shown in Fig. 3.20.

Fig. 3.21 demonstrates the effect of $Splice(aSym, b)$

A closed polygon bounded by edges $a_1, \ldots, a_n$ is constructed by applying *Splice* sequentially as in the following algorithm

$Polygon(a_1, a_2, ..., a_n)$ {
$Splice(a_1Sym, a_2)$; $Splice(a_2Sym, a_3)$; ...
$Splice(a_{n-1}Sym, a_n)$; $Splice(a_nSym, a_1)$;
}

An algorithm for removal of an edge a from a subdivision is as follows

$Remove(a)$ {
  $Splice(a, aOnext)$;
  $Splice(a, aSymOnext)$;

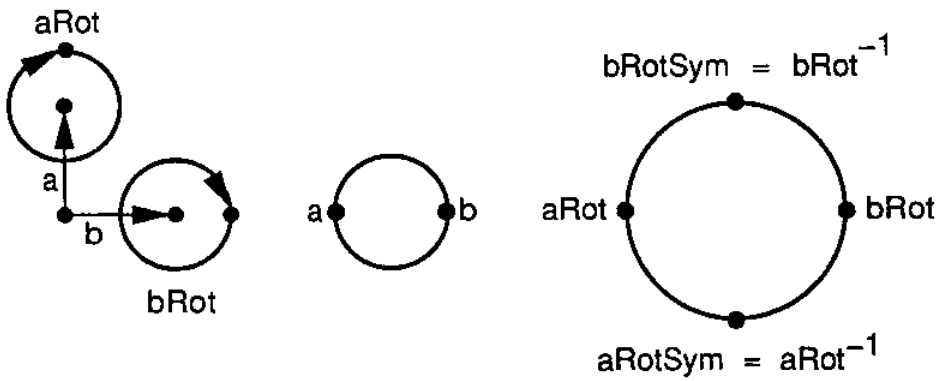

Figure 3.20: Edges combined by *Splice*

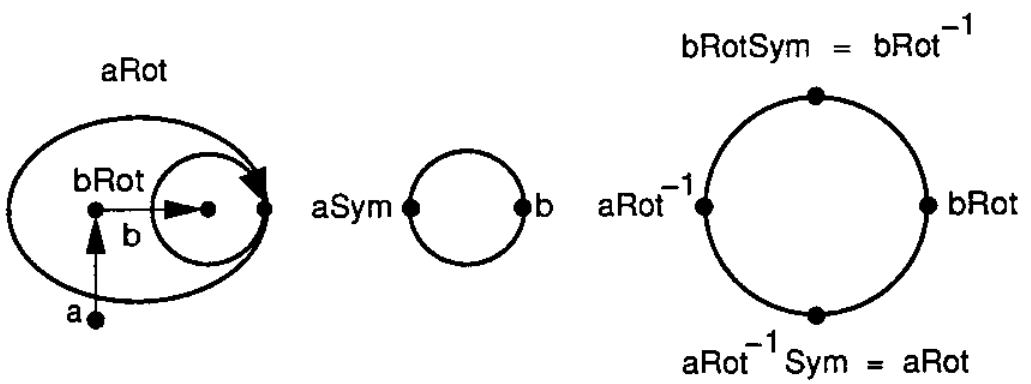

Figure 3.21: Edges combined by *Splice*

$Purge(a)$;
}

where $Purge(a)$ deletes the node associated with edge a from the structure.

An edge a is divided in two edges a_1, a_2 by the following algorithm

$Divide(a)$ {
$a_1 = Makedge()$;
$a_2 = Makedge()$;
$Splice(aOnext, a)$;
$Splice(aOnext, a_1)$;
$Splice(aSymOnext, aSym)$;
$Splice(aSymOnext, a_2)$;
$Splice(a_1Sym, a_2)$;
}

The quad-edge structure is used in [40] to provide a data structure for the construction of Voronoi diagrams or their duals, Delaunay triangulations. Nevertheless, it may be used as a basis of a B-rep modeler for two-manifold polyhedra as demonstrated by the above examples. It is worth mentioning that any high level operator can be constructed by using only two primitive operators compared to the multitude of Euler operators required by Euler-based modelers.

3.3.2 The Facet-Edge Pair Structure

The data structure by Dobkin and Laszlo [29, 30, 62] is the three-dimensional analogue of a two-manifold subdivision. The modeling space of the proposed data structure consists of three-manifold subdivisions or polyhedral subdivisions, $(\Re^3, C)$. Intuitively, a polyhedral subdivision is a collection of polyhedra incident along faces homeomorphic to disks (facets). A polyhedral subdivision may be drawn on the boundary of a four-dimensional sphere just as a polygonal or two-dimensional subdivision may be drawn on the boundary of a three-dimensional sphere. In the data structure of Dobkin and Laszlo, individual polyhedra should be orientable and of genus 0, i.e. homeomorphic to a 3-ball, so they are not allowed to have any handles or cavities. However, this does not pose a serious restriction of the modeling space. Polyhedra of a genus greater than 0 may always be decomposed into a collection of polyhedra homeomorphic to a 3-ball. Further, in solid modeling we are not, in general, interested in non-orientable polyhedra.

The basic element of the data structure is the *facet-edge pair* $a = (f, e)$ consisting of a facet f and an edge e on the facet's boundary. The facet of pair a is denoted by $facet(a)$, the edge by $edge(a)$. We shall use the notation $a \in C$ if $edge(a) \in C$, $facet(a) \in C$ and $edge(a), facet(a)$ are incident. If $E_f = (e^0 = e, e^1, \ldots, e^{n-1})$ is a circular ordering (ring) of edges incident to f (Fig. 3.22), such that e^{k-1}, e^k are adjacent, $0 \leq k \leq n-1$, then there exist two possible orientations of the facet f induced by E_f and E'_f, where E'_f contains the same edges in reverse order. One of these two rings is attached to the facet-edge pair a and is called the edge ring of a. Each orientation induces a direction on edge e, which allows the definition of vertices of origin and destination of the facet-edge pair, $aOrg, aDest$. In a similar manner, there exist a ring of faces F_e and its reverse, F'_e with edge e on their boundary. The ring F_e, which induces a sense of rotation around the edge e, is called the spin of the facet-edge pair a. One of the two above rings

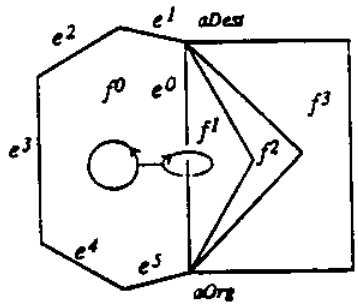

Figure 3.22: The handcuff diagram, adapted from Laszlo

of facets is attached to facet-edge pair a, and is called the facet ring of a. Facet f is incident to two polyhedra, *aPpos*, *aPneg*, such that if a stick normal to edge e rotates in the sense of *aSpin*, it first enters in polyhedron *aPneg*, then crosses facet f and enters in *aPpos*. Fig. 3.22 depicts the facet-edge pair by the so-called *handcuff* diagram.

There exist four oriented and spun facet-edge pairs sharing a facet f and edge e. They are a, *aSpin*, *aClock* and *aSpinClock*, illustrated in Fig. 3.23. The facet-edge pair *aEnext* has the same orientation, spin and facet component as a and its edge component is the next edge in E_a. Similarly, *aFnext* is defined as the facet-edge pair with same orientation, spin and edge component as a, and a face component co incident with the next face in facet ring F_a (Fig. 3.23). Functions *Spin*, *Clock*, *Fnext*, *Enext* have the following properties

$$aSpin^2 = a \tag{3.59}$$

$$aClock^2 = a \tag{3.60}$$

$$a(SpinClock)^2 = a \tag{3.61}$$

$$aFnext^{-1} = aClockFnextClock = aSpinFnextSpin \tag{3.62}$$

$$aEnext^{-1} = aClockEnextClock = aClockSpinEnextClockSpin \tag{3.63}$$

$$aClockFnext^i \neq a \ \forall i \tag{3.64}$$

$$aSpinFnext^i \neq a \ \forall i \tag{3.65}$$

$$aClockEnext^i \neq a \ \forall i \tag{3.66}$$

$$aSpinEnext^i \neq a \ \forall i \tag{3.67}$$

$$a \in C \Leftrightarrow aFnext \in C \tag{3.68}$$

Figure 3.23: The facet-edge pair functions

$$a \in C \Leftrightarrow aClock \in C \tag{3.69}$$

$$a \in C \Leftrightarrow aSpin \in C \tag{3.70}$$

The dual of a facet-edge pair $a = (f, e)$, denoted *aSdual*, is defined as the facet-edge pair consisting of the dual edge to facet f and the dual facet to edge e in the dual 3-complex $(\Re^3, C^*)$ such that $E_a Sdual$ and $F_a Sdual$ are composed of the duals of E_a, F_a. Formally,

$$edge(aSdual) = face(a)^* \tag{3.71}$$

$$face(aSdual) = edge(a)^* \tag{3.72}$$

$$E_a^* = (f_0^* = f^*, \ldots f_{n-1}^*) \tag{3.73}$$

$$F_a^* = (e_0^* = e^*, \ldots e_{n-1}^*) \tag{3.74}$$

where $E_a = (e_0 = e, \ldots, e_{n-1})$, $F_a = (f_0 = f, \ldots, f_{n-1})$. Recall that in a 3-complex duals of 1-cells, i.e. edges are 2-cells, i.e. facets and vice-versa. Fig. 3.24 shows a facet-edge pair and its dual. The rectangle is $facet(a)$ and its plane is normal to the page, the triangle is $facet(aSdual)$ and is located on the page. The duals of $aPpos, aPneg$ are vertices of $edge(aSdual)$. Function *Sdual* has the following properties

$$aSdual^2 = a \tag{3.75}$$

$$aClockSdual = aSdualClock \tag{3.76}$$

$$aSpinSdual = aSdualClockSpin \tag{3.77}$$

$$aEnext = aSdualFnextSdual \tag{3.78}$$

$$a \in C \Leftrightarrow aSdual \in C^* \tag{3.79}$$

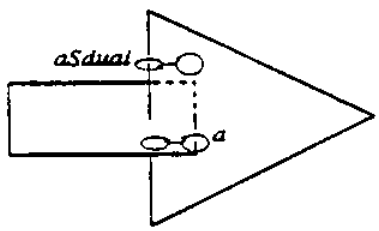

Figure 3.24: A facet-edge pair and its dual

the symmetric property $aFnext = aSdualFnextSdual$ can be proven easily using properties (3.75), (3.78).

A facet- edge algebra analogous to the edge algebra of [40] may be defined. The algebra is defined over a set of triplets (a, v, P), where a corresponds to a facet-edge pair, v to the origin of $edge(a)$ consistently with the orientation of a and p to $aPpos$. Functions *Spin, Clock, Enext, Fnext, Sdual* may be defined through the adjacency relationships in complex C and its dual C^* [62].

Function *aOrg* may be determined through a partition of the set of facet-edge pairs in complexes C, C^*. Thus, *aOrg* is an equivalence class of the above partition consisting of all facet-edge pairs with the same origin. Functions *Dest, Ppos, Pneg* are also partitions of the set of facet-edge pairs, which may be defined in terms of *Org* alone using the known functions *Clock, Spin, Sdual*, as follows

$$aDest = aClockOrg \tag{3.80}$$

$$aPneg = aSdualOrg \tag{3.81}$$

$$aPpos = aSdualDest \tag{3.82}$$

Functions *Dest, Pneg, Ppos* have, in addition, the following properties:

$$aOrg, aDest, aPpos, aPneg \text{ are all distinct} \tag{3.83}$$

$$aSpinOrg = aOrg \tag{3.84}$$

$$aSpinClockOrg = aDest \tag{3.85}$$

$$aFnext^iOrg = aOrg \ \forall \text{ integer } i \tag{3.86}$$

$$aSpinPpos = aPneg \tag{3.87}$$

$$aClockPpos = aPneg \tag{3.88}$$

$$aSpinClockPpos = aPpos \tag{3.89}$$

$$aEnext^{i}Ppos = aPpos \ \forall \text{ integer } i \tag{3.90}$$

Function *Srot*. analogous to *Rot* of the quad-edge structure. is defined as

$$aSrot = aSdualSpin = aSpinClockSdual \tag{3.91}$$

The facet-edge pair *aSrot* has a spin opposite to the spin of a and its edge is directed from *aPneg* towards *aPpos*. The following relations may be proven using the properties of *Clock*. *Spin*, *Sdual* (eqs. (3.59)-(3.79).

$$aSrot^2 = aClock \tag{3.92}$$

$$aSrot^3 = aSrot^{-1} \tag{3.93}$$

$$aSrot^4 = a \tag{3.94}$$

$$aSrot^2Fnext = aFnext^{-1}Clock \tag{3.95}$$

$$aSrotFnextSrot = aEnext^{-1}Clock \tag{3.96}$$

$$aSrot^3FnextSrot = aEnext \tag{3.97}$$

The data structure for a polyhedral subdivision consists of one node n for each facet-edge pair. Each node is an array of length 4. storing data pertaining to $aSrot^rSpin^s, r = 0,1,2,3, s = 0,1$. The elements of the array are structures with two fields. *Data* and *Next*. In the *Data* field geometrical and other non-topological information is stored. The *Next* field is a pointer to $aSrot^rFnext$. Further, a bit containing the value of s is assigned to each array element.

If we represent the expression $aSrot^rSpin^s$ by a triplet (n,r,s), then the above data structure captures all basic facet-edge pair functions as implied by the following relations.

$$(n,r,s)SRot = (n,r+12s,s) \tag{3.98}$$

$$(n,r,s)Spin = (n,r,s+1) \tag{3.99}$$

$$(n,r,s)Clock = (n,r+2,s) \tag{3.100}$$

$$(n,r,s)Sdual = (n,r+1+2s,s+1) \tag{3.101}$$

$$(n,r,s)Fnext = (n_1,2s,s) \text{ with } n_1 = n[r+2s].Next \tag{3.102}$$

$$(n,r,s)Fnext^{-1} = (n,r,s)ClockFnextClock \tag{3.103}$$

$$(n,r,s)Enext = (n,r,s)SdualFnextSdual \tag{3.104}$$

$$(n,r,s)Enext^{-1} = (n,r,s)ClockEnextClock \tag{3.105}$$

Operations involving r are computed mod 4, those involving s only are computed mod 2. The above relations may be proven using eqs. (3.59)-(3.63), (3.75)-(3.78). The correctness of the implementation may be demonstrated by proving eqs. (3.59)-(3.63), (3.75)-(3.78) using eqs. (3.98)-(3.105). However, it should be noted that eqs. (3.59)-(3.70) and (3.79) are not automatically guaranteed, therefore it is up to the basic operators to enforce them. The *Org* function, which is a partition of the set of facet-edge pairs, may be implemented by any suitable structure, such as a tree, list or hash table. Functions *Dest, Pneg, Ppos* are well-defined by eqs. (3.80)-(3.82).

A small number of basic operators may be used to construct any polyhedral subdivision. Operator *make_facet_edge*() returns a node corresponding to a single facet-edge pair, a. For any facet-edge pair $a' = aSdual^d Clock^c Spin^s, d, c, s \in \{0, 1\}$, the relation $a'Fnext = a'Enext = a'$ holds. Operator *splice_facets*(a, b) combines the facet rings F_a, F_b, if they are distinct, and, if they are identical, splits the common ring into two distinct rings. If $\alpha = aFnextClock, \beta = bFnextClock$, the effect of this operator is formalized by the following equations

$$aFnext \leftarrow bFnext \tag{3.106}$$

$$bFnext \leftarrow aFnext \tag{3.107}$$

$$\alpha Fnext \leftarrow \beta Fnext \tag{3.108}$$

$$\beta Fnext \leftarrow \alpha Fnext \tag{3.109}$$

$$aClockSpinFnext \leftarrow \beta Spin \tag{3.110}$$

$$bClockSpinFnext \leftarrow \alpha Spin \tag{3.111}$$

$$\alpha ClockSpinFnext \leftarrow bSpin \tag{3.112}$$

$$\beta ClockSpinFnext \leftarrow aSpin \tag{3.113}$$

The above assignments are assumed to be carried out in parallel, or alternatively, expressions on the right hand side of the above assignments may be considered as copies of the relevant entities stored in some auxiliary memory locations before serial execution of (3.106)-(3.113).

A dual operator to *splice_faces*(a, b) is *splice_edges*(a, b). Edge rings E_a, E_b of the facet-edge pair undergo analogous transformations under *splice_edges*(a, b) as F_a, F_b under *splice_faces*(a, b). Operator *splice_edges*(a, b) is implemented as

$$splice\_edges(a, b) = splice\_facets(aSdual, bSdual) \tag{3.114}$$

Operator *transfer* is used to manipulate partitions of the set of facet-edge pairs, such as $aOrg$, $aDest, aPpos, aPneg$. The operation $transfer(A, B)$ merges the equivalence classes A, B by transferring each element $b \in B$ into A.

Individual polyhedra of the polyhedral subdivision may be constructed and transformed by a data structure equivalent to the quad-edge structure of [40], which is derived from the structure for polyhedral subdivision, described previously. An edge of a polyhedron p may be represented by a pair $< a, d >$, where a is a facet-edge pair and d a duality bit. The edge functions are defined as follows

$$< a,0 > Flip =< aFnextSpin, 0 > \tag{3.115}$$

$$< a,0 > Sym =< aFnextClock, 0 > \tag{3.116}$$

$$< a,0 > Onext =< aEnext^{-1}FnextClock, 0 > \tag{3.117}$$

$$< a,d > Dual =< a, 1-d > \tag{3.118}$$

$$< a,1 > Flip =< aSpinClock, 1 > \tag{3.119}$$

$$< a,1 > Sym =< aFnextClock, 1 > \tag{3.120}$$

$$< a,1 > Onext =< aEnext^{-1}, 1 > \tag{3.121}$$

$$< a,d > Rot =< a,d > FlipDual \tag{3.122}$$

The validity of eqs. (3.21)-(3.30), (3.32)-(3.42) of the edge algebra may be shown using eqs. (3.59)-(3.70) and (3.80)-(3.90). Thus, if A is the set of facet-edge pairs of the polyhedron P, $E = \{< a,0 >, a \in A\}$, $E^* = \{< a,1 >, a \in A\}$ and $Onext, Rot, Flip$ as defined above, an edge algebra is obtained and operators *Makedge* and *Splice* of [40] may be used to manipulate single polyhedra.

In [62], the operators make_segment(), make_loop() simulating *Makedge* are implemented. These operators make use of *make_facet_edge, splice_facets* and *splice_edges*. Operator *make_segment* creates a subdivision of the sphere consisting of a single edge and *make_loop* its dual. Operator *Splice* of [40] is simulated by calls to *splice_edges*. Two polyhedra P_a, P_b, prepared by *make_segment, make_loop* and the facet-edge version of splice may be glued together by operator $meld(a,b)$, where a, b are facet-edge pairs of P_a, P_b, respectively, such that $facet(a) \in \partial P_a$, $facet(b) \in \partial P_b$ and the edge rings E_a, E_b consist of the same number of edges, l. This operator glues *aPneg, bPpos* along $facet(a), facet(b)$ by fusing $edge(a_i), edge(b_i)$, where $a_i = aEnext^i, b_i = bEnext^i, 0 \leq i \leq l-1$. If the two polyhedra belong to two distinct polyhedral subdivisions $(\Re^3, C_a), (\Re^3, C_b)$, these are combined into one subdivision by the *meld* operator.

Applications discussed in [29, 30, 62] include construction of three-dimensional Voronoi diagrams and Delaunay triangulations, construction of weighted Voronoi diagrams, and decomposition and manipulation of 4-polyhedra.

3.3.3 The Hexblock Structure

In [17], Buckley reports a divide-and-conquer algorithm for computing 4-dimensional convex hulls, which uses the *hexblock* data structure. This data structure is capable of describing three-dimensional subdivisions. However, no detailed description is given in [17] other than a figure illustrating connections of topological elements with pointers. The author states that storage complexity is linear with respect to the number of edges and time complexity for calculating topological adjacencies is also linear with respect to the size of the output set. Further, his splice operators work safely, so there is no danger of producing inconsistent subdivisions by inept usage of the basic constructors as alluded in [29]. It is mentioned in [17] that a paper describing the hexblock structure is in preparation.

3.3.4 The Cell-Tuple Structure

Brisson [12, 13, 14] presented a data structure for describing subdivisions of n-manifolds, $(\Re^n, C)$ called *cell-tuple* structure. The quad-edge structure [40] for subdivision of two-manifolds and the facet-edge pair structure [29] for subdivisions of three-manifolds may be represented by appropriate specialization of the cell-tuple structure. No restrictions are posed to the underlying n-manifold which may be a closed surface embedded in $\Re^{n+1}$ or a manifold with boundary. The cell-tuple structure makes use of the incidence graph whose nodes are the cells of the set C and the *switch* function described below.

The cells of $C = \{c_\alpha\}_{\alpha \in A}$ are partially ordered by a relation "$<$" defined as $c_{\alpha_1} < c_{\alpha_2}$ if $c_{\alpha_1}, c_{\alpha_2}$ are incident and $c_{\alpha_1} \subseteq \partial c_{\alpha_2}$. If, in addition, $\dim c_{\alpha_2} = \dim c_{\alpha_1} + 1$ we shall write $c_{\alpha_1} \langle c_{\alpha_2}$. Set C is assumed to contain two unique abstract cells, c_{-1} of dimension -1 and c_{n+1} of dimension $n+1$, which are incident to all cells of C, i.e. $c_{-1} < c_\alpha < c_{n+1}$, $\alpha \in A$. Given two incident cells $c_{\alpha_1}, c_{\alpha_2}$, with $c_{\alpha_1} < c_{\alpha_2}$, the set of cells incident to both $c_{\alpha_1}, c_{\alpha_2}$ and of dimension higher than that of c_{α_1} and lower than that of c_{α_2} is denoted by $S(c_{\alpha_1}, c_{\alpha_2})$. Thus,

$$S(c_{\alpha_1}, c_{\alpha_2}) = \{c \in C,\ c_{\alpha_1} < c < c_{\alpha_2}) \tag{3.123}$$

A cell-tuple is a tuple $t = (c_{\alpha_0}, c_{\alpha_1}, \ldots, c_{\alpha_n})$, where c_{α_k} is a cell of dimension k , $c_{\alpha_{k-1}} \langle c_{\alpha_k}$, $1 \leq k \leq n$. The kth element of the cell-tuple is denoted by t_k. Further, T_M is the set of all cell-tuples of subdivision (M, C).

$$T_M = \{t = (c_{\alpha_0}, \ldots, c_{\alpha_n}),\ c_{\alpha_k} \in C\ ,\ 0 \leq k \leq n\ ,\ c_{\alpha_{k-1}} \langle c_{\alpha_k}\} \tag{3.124}$$

In [12, 13, 14] the following Lemma provides the basis for the definition of the *switch* operator.

Lemma 1 For every $t \in T_M$ there is a unique $t' \in T_M$ such that $t'_k \neq t_k$ and $t'_i = t_i \quad \forall i \neq k, 0 \leq k \leq n$. In other words, given any cell-tuple, there is a unique cell-tuple with all but one identical cells.

The $switch_k$ operator is a function $switch_k : T_M \rightarrow T_M$, which when applied to a tuple t delivers tuple t' defined as in the above Lemma. A variant of Lemma 1 is as follows.

Lemma 1a Given a subdivision (M, C) of an n-manifold M and cells $c_{\alpha_{k-1}} \langle c_{\alpha_k} \langle c_{\alpha_{k+1}}$ of dimensions $k-1, k, k+1$, respectively, $0 \leq k \leq n$, there is a unique cell $c'_{\alpha_k} \in C, c'_{\alpha_k} \neq c_{\alpha_k}$, such that $c_{\alpha_{k-1}} \langle c'_{\alpha_k} \langle c_{\alpha_{k+1}}$. The cell-tuple structure is a pair $\vec{T}_M = \{T_M, \{switch_k\}, 0 \leq k \leq n\}$. The *switch* operator provides a tool for navigating through the incidence graph of the subdivision. Based on *switch*, a family of switch-like functions is constructed as follows. Function $switch_{kl} : T_M \rightarrow T_M$ is defined as $switch_l(switch_k(t))$. In general, if $w = w_1 \ldots w_l \in \{0, \ldots, n\}*$, then

$$switch_w(t) = \begin{cases} switch_{w_l}(switch_{w_{l-1}}(\ldots(switch_{w_1}(t))\ldots)) \\ t \text{ if } w \text{ is the empty word} \end{cases} \tag{3.125}$$

The star notation used above is explained here briefly. If A is a finite set, the alphabet, words based on alphabet A are strings consisting of elements of set A, $a_l a_{l-1} \ldots a_0, a_i \in A$, $0 \leq i \leq l$ in the same manner words in any language are built from letters. Any element may be repeated more than one times in a word. The empty string is the empty word. The symbol $A*$ is used

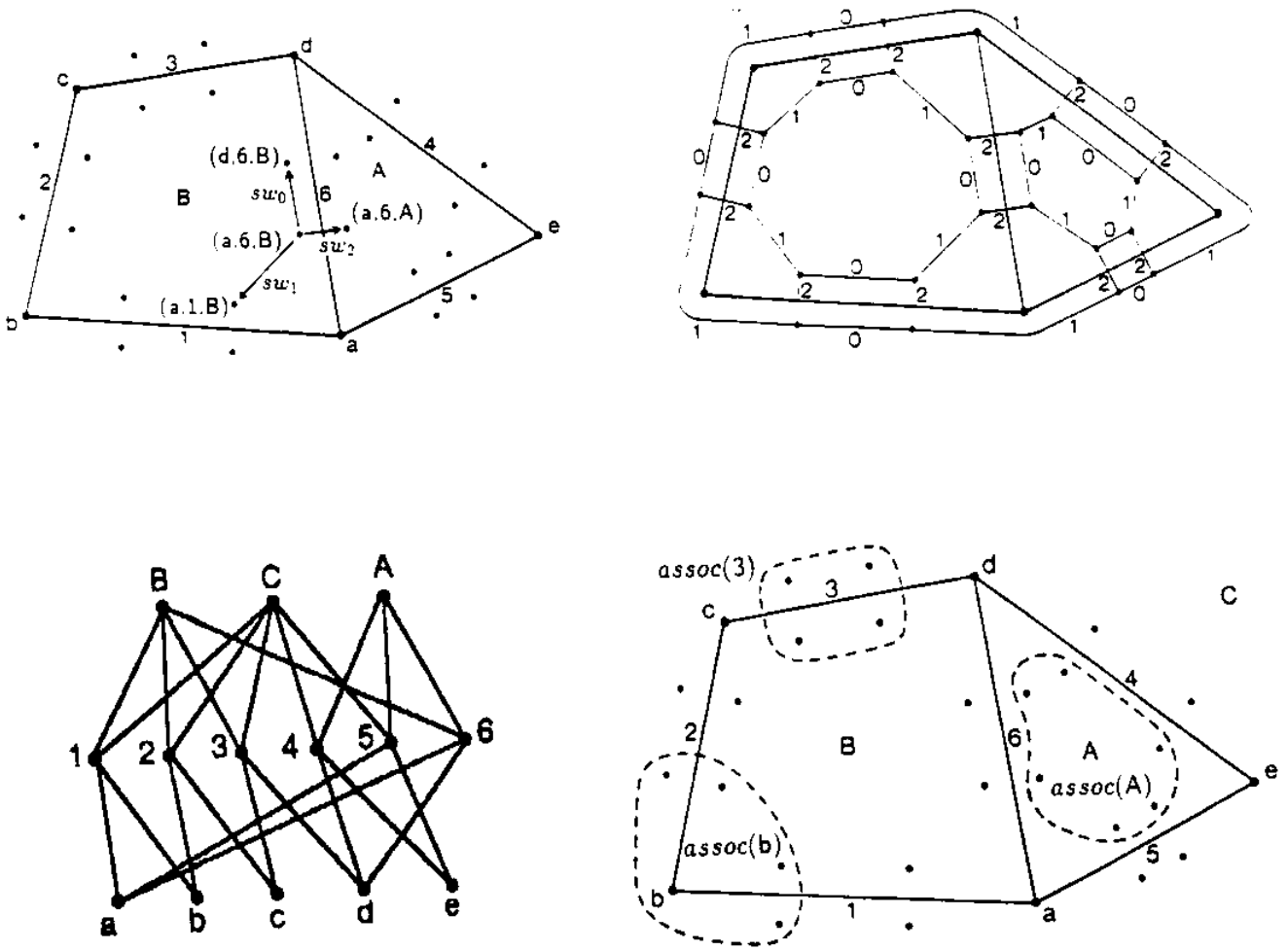

Figure 3.25: The cell-tuple structure and the incidence graph, adapted from Brisson

to characterize the set of words constructed from alphabet A. Thus, the index w in (3.125) may contain the same digit several times.

Fig. 3.25 shows a plane figure (subdivision of the sphere), where cell-tuples are represented by small dots. The dot corresponding to a tuple (v, e, f) is inside the face f and nearer to vertex v and edge e than any other dot. The *switch* function may be visualized by a labeled graph G_M (Fig. 3.25) with cell-tuples as nodes and arcs connecting tuples associated by some *switch* function. The arc is labeled with k.

Given a set $I \subseteq \{0, \ldots, n\}$, the I-orbit of a tuple t, denoted $switch_{I*}(t)$, is defined as

$$switch_{I*}(t) = \{t' \in T_M,\ t' = switch_w(t) \forall w \in I*\} \tag{3.126}$$

The I-orbit of a tuple t consists of all tuples that may be reached from t in graph G_M by following paths in G_M, whose arcs are labeled by elements of set I.

For any cell $c \in C$ with $dim(c) = k$, the set of associated cell-tuples is defined as

$$assoc(c) = \{t \in T_M,\ t_k = c\} \tag{3.127}$$

In Fig. 3.25, the *assoc* sets of a face, edge and vertex are shown. Also shown is the incidence graph of the subdivision, I_M.

Two cell-tuple structures $\vec{T}_M, \vec{T}_N$ are equivalent if there is a bijection $f : \vec{T}_M \rightarrow \vec{T}_N$ such that $switch_k(f(t)) = f(switch_k(t)) \forall t \in T_M$. In [12, 13, 14], two important theorems are proven

Theorem 1 If $(M, C), (N, D)$ are subdivided n-manifolds, then the following are equivalent:

1. $(M, C), (N, D)$ are equivalent
2. The incidence graphs of the subdivisions, I_M, I_N are isomorphic;
3. The cell-tuple structures $\vec{T}_M, \vec{T}_N$ are equivalent.

Theorem 2 Given two incident cells $c_{\alpha_{k-2}}, c_{\alpha_{k+1}} \in C$, $c_{\alpha_{k-2}} < c_{\alpha_{k+1}}$ of dimensions $k-2, k+1$ respectively, there exists a cell-tuple $t^0 \in \vec{T}_M$ such that $t^0_{k-2} = c_{\alpha_{k-2}}, t^0_{k+1} = c_{\alpha_{k+1}}$. Further, for any such t^0, if the sequence of cell-tuples $t^0, \ldots, t^{m-1}$, where m is the number of elements of $S(c_{\alpha_{k-2}}, c_{\alpha_{k+1}})$ according to (3.123) is defined by

$$t^i = \begin{cases} switch_{k-1}(t^{i-1}) & i \text{ even} \\ switch_{k-1}(t^{i-1}) & i \text{ odd} \end{cases} \tag{3.128}$$

then the cell sequence $c_{\eta_0}, \ldots c_{\eta_{m-1}}$ defined by

$$c_{\eta_i} = \begin{cases} t^i_{k-1} & i \text{ even} \quad 2 \leq i \leq m \\ t^i_k & i \text{ odd} \quad 1 \leq i \leq m-1 \end{cases} \tag{3.129}$$

provides a circular ordering of $S(c_{\alpha_{k-2}}, c_{\alpha_{k+1}})$.

An alternative formulation of the same theorem is as follows.

Theorem 2a Under the same assumptions of Theorem 2 there exist cells $c_{\eta_0}, c_{\eta_1} \in C$ such that $c_{\alpha_{k-2}} \langle c_{\eta_0} \langle c_{\eta_1} \langle c_{\alpha_{k+1}}$. For any such c_{η_0}, c_{η_1} the sequence $c_{\eta_0}, \ldots c_{\eta_{m-1}}$ defined by

$$c_{\eta_i} = \begin{cases} switch_{k-1}(c_{\alpha_{k-2}}, c_{\eta_{i-2}}, c_{\eta_{i-1}}) & i \text{ even} \quad 2 \leq i \leq m \\ switch_k(c_{\eta_{i-1}}, c_{\eta_{i-2}}, c_{\alpha_{k+1}}) & i \text{ odd} \quad 1 \leq i \leq m-1 \end{cases} \tag{3.130}$$

gives a circular ordering of $S(c_{\alpha_{k-2}}, c_{\alpha_{k+1}})$.

Theorem 1 provides the theoretical basis for the unambiguous description of subdivided n-manifolds either by the cell-tuple structure or by the incidence graph. With the help of Theorem 2 a circular ordering of cells adjacent to a given cell may be defined. For example, in a three-dimensional subdivision, the edges and faces of a shell s adjacent to a vertex v may be ordered as follows. Find an edge e_1 and a face f_1 of the shell incident to the vertex and to each other. Then, find the unique edge e_2 of this face, which shares with the above edge the vertex v, i.e. $(v, e_2, f_1, s) = switch_1(v, e_1, f_1, s)$. Proceed by locating a face f_2 of shell s containing edge e_1, i.e. $(v, e_2, f_2, s) = switch_2(v, e_2, f_1, s)$. The procedure is repeated until we return to edge e_1.

Theorems 1 and 2 as well as Lemma 1 providing the definition of the *switch* function are proven by refining (M, C) through a generalized barycentric subdivision. The dual of a subdivision may be constructed by simply turning the incidence graph upside down. Thus, vertices of the primal graph are n-dimensional cells of the dual. In general, the dual of a k-cell is an $(n-k)$-cell and incidence relations are preserved. If we represent the canonical map of the dual transformation by $dual : C \to C^*, c^*_\alpha = dual(c_\alpha)$, the set of dual cell-tuples is defined as

$$T^*_M = \{(c^*_{\alpha_n}, \ldots, c^*_{\alpha_0}), (c_{\alpha_0}, \ldots, c_{\alpha_n}) \in T_M\} \tag{3.131}$$

and the dual *switch* function as

$$switch^*_k(c^*_{\alpha_n}, \ldots, c^*_{\alpha_k}, \ldots, c^*_{\alpha_0}) = (c^*_{\alpha_n}, \ldots, c^{*\prime}_{\alpha_k}, \ldots, c^*_{\alpha_0}) \tag{3.132}$$

where

$$c^{*\prime}_{\alpha_k} = switch_{n-k}(c_{\alpha_0}, \ldots, c_{\alpha_n})^* \tag{3.133}$$

In [12, 13, 14] the dual *switch* function is defined as follows

$$switch_R(c_{\alpha_0}, \ldots, c_{\alpha_n}) = (c^*_{\alpha_n}, \ldots, c^*_{\alpha_0}) \tag{3.134}$$

The *switch* function possesses some basic properties. Let $t \in T_M \cup T^*_M$, $i \neq j, i, j \in \{0, \ldots, n\}$. Then

$$switch_i(t) \neq t \tag{3.135}$$

$$switch_{ij}(t) \neq t \tag{3.136}$$

$$switch_{i^2}(t) = t \tag{3.137}$$

$$\text{if } j = i \pm 1,\ \exists m \geq 2 \text{ such that } switch_{(ij)^m}(t) = t \tag{3.138}$$

$$\text{if } j \neq i,\ i \pm 1, \text{ then } switch_{(ij)^2}(t) = t \tag{3.139}$$

$$switch_{RiR}(t) = switch_{n-i}(t) \tag{3.140}$$

Although the dual subdivision is automatically represented by the cell-tuple structure, it does not enhance its capabilities. Function $switch_R$ is defined mainly to establish equivalence of the cell-tuple structure with the quad-edge structure and the structure based on facet-edge pairs. Therefore, we may restrict ourselves to properties (3.135)-3.139), which hold for tuples of the primal subdivision, i.e. $t \in T_M$.

So far the cell-tuple structure for subdivided n-manifolds *without boundary* has been discussed. If M is an n-manifold *with boundary*, it may be shown that $(\partial M, C^\partial)$, where $C^\partial = \{c \in C,\ c \subseteq \partial M\}$ is a subdivided $(n-1)$-manifold without boundary. We add an extra abstract cell c_{α_∞} to set C, where $\dim(c_{\alpha_\infty}) = n$ and $\forall c \in C^\partial c < c_{\alpha_\infty}$. It should be noted that the complement of M with respect to $\Re^n$ is not always a manifold, hence the predicate 'abstract' to c_{α_∞}. If $C^+ = C \cup \{c_{\alpha_\infty}\}$ then Lemma 1 providing the uniqueness of $switch_k(t)$ still holds for (M, C^+). Therefore, with the

addition of the abstract cell c_{α_∞} subdivisions of manifolds with boundary may be accommodated in the cell-tuple structure.

In subdivided two-manifolds a cell-tuple (t_0, t_1, t_2) may be associated with a unique edge reference of the quad-edge structure. The direction of edge t_1 $(e = t_1)$ is implied by vertex $t_0(t_0 = eOrg)$ and its orientation by face $t_2(t_2 = eLeft)$. Thus, a bijection ϕ between the cell-tuple set T_M and the edge set E may be established, $\phi : T_M \rightarrow E$. Function $switch_{12}$ corresponds to *Onext*, function $switch_2$ to *Flip* and function $switch_{2R}$ to *Rot*, or equivalently, if $e = \phi(t)$, then

$$eOnext = \phi(switch_{12}(t)) \tag{3.141}$$

$$eFlip = \phi(switch_2(t)) \tag{3.142}$$

$$eRot = \phi(switch_{2R}(t)) \tag{3.143}$$

Further,

$$eSymFlip = \phi(switch_0(t)) \tag{3.144}$$

$$eOnextFlip = \phi(switch_1(t)) \tag{3.145}$$

All basic properties of the edge algebra, eqs. (3.21)-(3.30) may be proven by using eqs. (3.135)-(3.140) and, conversely, if function $switch_k$, $k = 0, 1, 2$ is defined by (3.144), (3.145) and (3.142), respectively, properties (3.21)-(3.30) of the edge algebra may be used to establish (3.135)-(3.140). The completion of a two-manifold subdivision used by Guibas and Stolfi in [40] to prove equivalence between their edge algebra and two manifold subdivisions is a specialization of the generalized barycentric subdivision.

Similarly, a tuple (v, e, f, s) of a three-manifold subdivision is associated with a unique facet-edge pair a. The orientation of a can be determined uniquely through vertex v, which induces a direction on edge e and hence an orientation on face f. The spin of a is determined by shell s, which by convention may be set equal to *aPpos*. Operators *Clock*, *Enext*, *Fnext*, *Rev* and *Sdual* may then be expressed in terms of appropriate *switch* functions. All basic constructions given in [40, 62] may be formulated in the cell-tuple structure in terms of repeated applications of the $switch_w$ function, where the length of word w is at most 2.

In addition, special objects like *edgeuses*, *faceuses* in Weiler [112] and *cusps* in Choi [25] correspond to *orbits* in the cell-tuple structure.

One basic operator defined in [12, 13, 14] for the manipulation of subdivisions is $attach_k$. This operator modifies the cell-tuple structure by associating the kth component of two cell-tuples t^1, t^2 with each other. The formal definition of $attach_k$ is as follows

$$attach_k(t^1, t^2) : (T_M, switch) \rightarrow (T_M, switch') \ , \ t^2 \neq switch_k(t^1) \tag{3.146}$$

$$switch'_k(t^1) = t^2 \tag{3.147}$$

$$switch'_k(t^2) = t^1 \tag{3.148}$$

$$switch'_k(switch_k(t^1)) = switch_k(t^2) \tag{3.149}$$

$$switch'_k(switch_k(t^2)) = switch_k(t^1) \tag{3.150}$$

$$switch'_k(t) = switch_k(t),\ \forall t \in T_M,\ t \neq t^1, t^2 \tag{3.151}$$

It should be noted that the above operator satisfies conditions (3.135,3.137) but in certain circumstances conditions (3.138,3.139) may not be fulfilled. Another basic operator, $genljoin_k$, satisfies all conditions (3.135)-(3.139) and hence produces valid cell-tuple structures. Operator $genljoin_k$ produces a new cell-tuple structure by applying $attach_k(switch_w(t^1), switch_w(t^2))$ for all words $w \in I*$, where $I = \{0, \ldots n\} - \{k-1, k, k+1\}$.

Brisson discusses four implementation methods for data organization in the cell-tuple structure. The first is the graph approach, based on the incidence graph. Any appropriate data structure for representation of graph structures may be used. A convenient method based on the hierarchical structure of the incidence graph is to store incidence relationships between cells of dimensions differing by one. Either direction from higher to lower dimensional cells or both may be captured in the data structure. The set of standard operators in the graph structure may include *make_node* to generate a node representing a cell, *kill_node* to destroy a node, $make\_incident(\nu_1, \nu_2)$ to make nodes ν_1, ν_2 incident provided that $\dim\nu_2 = \dim\nu_1 + 1$ and $kill\_incident(\nu_1, \nu_2)$ to remove an arc from the graph.

The second implementation method, the data base approach, uses a data base management system to store all tuples of an n-dimensional subdivision. To each cell c_{α_k} of a tuple $t = (c_{\alpha_0}, \ldots c_{\alpha_n})$ a pointer is attached indicating the location of $switch_k(t)$ in the data base.

The third implementation method involves storing triples $(c_{\alpha_{k-1}}, c_{\alpha_k}, c_{\alpha_{k+1}})$ instead of entire tuples and associate to each triple a single pointer to triple $(c_{\alpha_{k-1}}, c'_{\alpha_k}, c_{\alpha_{k+1}})$ according to the second form of Lemma 1. The second and third approaches guarantee consistency and correctness of the cell-tuple structure.

The fourth implementation method, the pointer approach, uses a record consisting of an array of pointers to cell descriptors of the cell-tuple components and another array of pointers to cell-tuple records obtained by applying the switch operator.

In [12, 13, 14], an analysis of storage complexity for three standard subdivisions, the minimal subdivision of an n-sphere, the n-simplex and the n-dimensional hypercube is carried out.

In any implementation of the cell-tuple structure a set of basic constructors for incremental building of subdivisions should be present. Constructor *make_vertex* produces a vertex representing a minimal subdivision of a zero-manifold. Constructor *kill_vertex* removes an isolated vertex. Constructor *lift* (Fig. 3.26) accepts as input a subdivision (M, C) of an $(n-1)$-manifold M and produces a new subdivision $M', C \cup \{c_{\alpha_n}\}$, where M' is an n-manifold homeomorphic to $\mathrm{cl}(B^n)$ and $\partial c_{\alpha_n} = C$. In other words, c_{α_n} is the n-dimensional cell, whose boundary consists of all cells of subdivision (M, C). Operator *unlift* has the reverse effect.

Constructor *join* merges two cells c_α, c'_α which should be equivalent if they are considered as subdivided manifolds together with their boundaries. If the *join* constructor is applied to n-manifold subdivisions, cells c_α, c'_α should be $(n-1)$-cells. In addition, they should be boundary cells in subdivisions of manifolds with boundary. The parameters passed to *join* include pairs of identified boundary cells of c_α, c'_α, so that the merging operation is defined in an unambiguous way. One method is to identify pairs of vertices to be merged in one. Another method is to specify chains $(c_{\alpha_0}, \ldots c_{\alpha_k} = c_\alpha)$, $(c'_{\alpha_0}, \ldots c'_{\alpha_k} = c'_\alpha)$, such that $c_{\alpha_{i-1}}(c_{\alpha_i}$, $c'_{\alpha_{i-1}}(c'_{\alpha_i} 1 \leq i \leq k$. Constructor

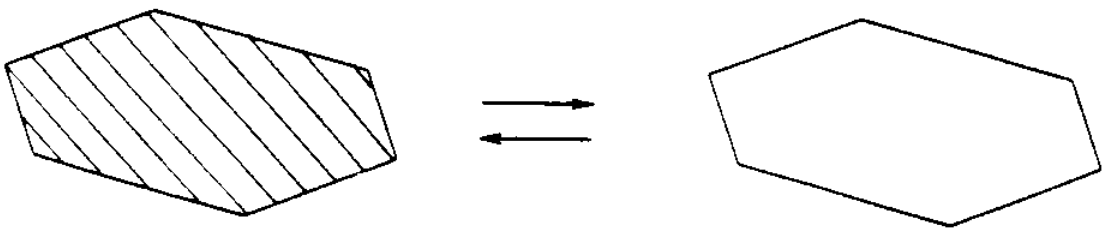

Figure 3.26: The *lift* and *unlift* operator, adapted from Brisson

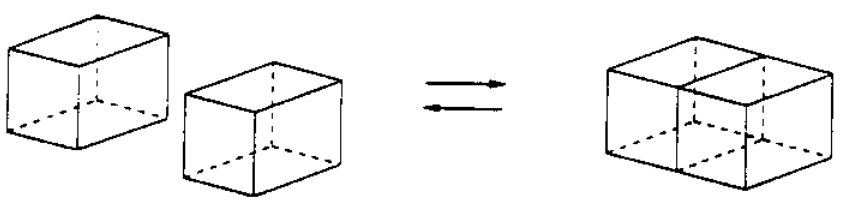

Figure 3.27: The *join* and *unjoin* constructor, adapted from Brisson

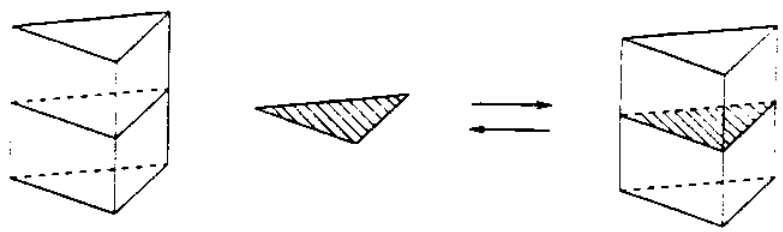

Figure 3.28: The *split* and *unsplit* constructor, adapted from Brisson

join will then merge $(c_{\alpha_i}, c'_{\alpha_i})$ in one cell. The remaining boundary cells of c_α, c'_α are identified by applying the *switch* operator. The reverse constructor is *unjoin*. Fig. 3.27 illustrates the effect of *join* and *unjoin*.

The *split* constructor is used to divide k-cells and provides a generalization of Euler operators. It accepts as input a number of cells on the boundary of a cell c_{α_k} such that their union is homeomorphic to sphere S^{k-2}. A $(k-1)$-cell is built first by applying the lift constructor to the above set and their cell c_{α_k} is split into two cells (Fig. 3.28). The reverse constructor, *unsplit*, eliminates a cell of dimension $k-1$, which is incident to two k-cells only and merges those cells in one.

Any subdivision of an n-manifold may be built incrementally by a repeated application of the basic constructors. However, simple constructors for generating simple subdivisions, such as standard geometrical figures, may be easily implemented. For example, in the incidence graph approach, all one has to do is to provide the incidence graph in the format required by the particular implementation of the data structure.

In [12, 13, 14] details of an implementation are given based on the data base approach. The data structure is centered around nodes of two types, the cell-tuple node and the cell-desc node. The cell-tuple node contains an array of pointers, one pointer for each one of its cells pointing to other cell-tuples associated to it by *switch*. Another field of the cell-tuple node is an array of pointers to cell-desc nodes describing the geometry of each one of its cells. Other fields contain the dimension of the tuple and auxiliary data used in the implementation of the construction and basic interrogation algorithms. The data structure is implemented in C++ allowing a unified treatment of geometrical specifications by a hierarchical organization of geometrical primitives.

3.3.5 Other Abstract Approaches

Lienhardt [65, 66] has developed a topological model for the description of subdivisions of either orientable or non-orientable n-manifolds with or without boundary. The model uses combinatorial objects, the $n-G$-maps, consisting of involutions operating on a set B, whose elements are called *darts*. An involution α is a function $\alpha : B \rightarrow B$ with $\alpha^2(b) = b \; \forall \; b \in B$. A dart may be regarded as one half of an edge including one of its vertices. It may be shown that there is a one-to-one correspondence between $n-G$-maps and abstract cell-tuple structures obeying (3.135), (3.136), (3.137) and (3.139). The dual subdivision may be easily constructed from an $n-G$-map describing the primal subdivision. For some models discussed so far [7, 29, 12, 13, 14, 40, 108], an equivalent $n-G$-map is sketched. Some basic construction operators and a suitable data structure for $n-G$-maps are discussed in [66]. An implementation of this model is described in [31].

In [77] the Winged Representation for modeling of n-dimensional, regular simplicial complexes (**Definition 19a**) is introduced. The complex may be embedded in a higher dimensional space. The data structure is very simple. For every n-dimensional simplex σ_k of the complex, a set of $n+1$ pointers is stored, each pointer pointing to the simplex $\sigma_{k,l}$ adjacent to σ_k along its l-th face (it is recalled that a n-simplex has $n+1$ faces of dimension $n-1$ each). In the case of a simplicial complex with boundary some pointers associated with boundary faces of the complex have a 0 (or NULL) value. Another set of $n+1$ pointers points to the vertices of simplex σ_k, which bear the geometric information, i.e. vertex coordinates in the embedding space.

Useful operators for the creation and transformation of simplicial complexes are defined and implemented. Such operators are affine transformation (translations, rotations, scaling, shear), the boundary operator, sweeping, extrusion, projection and set operators.

Affine transformations are simple geometrical operations on the coordinates of the vertices of the complex. Given a n-dimensional, regular, simplicial complex P with boundary, the boundary operator creates the Winged Representation of the $n-1$-dimensional complex ∂P consisting of boundary faces of P. To derive the new structure, a consistent orientation and numbering scheme of the boundary faces of a n-simplex based on permutations of vertex indices is used.

When a n-dimensional simplicial complex P embedded in $\Re^N$ is swept, a higher dimensional complex embedded in $\Re^N$ results, whose dimension is $n+1$. Extrusion of P in $\Re^N$ creates a $n+1$-dimensional complex embedded in $\Re^{N+1}$. Projection of an n-dimensional complex embedded in $\Re^N$ generates a new complex of dimension $n-1$ embedded in $\Re^{N-1}$ by simply removing the same coordinate from all vertices.

Set operators (union, intersection, difference) are implemented using regular sets as in a CSG context. Geometrical operations (intersections) are performed on constituent simplexes. A simplicial decomposition of the resulting complex is generated. Since simplexes are convex sets, linear programming methods can be used to reduce complexity of geometrical operations. However, it should be noted that the complexity of set operations increases with n!, where n is the dimension of the complexes involved.

The extrusion, projection and set operators can be used for polyhedral approximation of free configuration spaces of mobile systems and in modeling vector fields varying with time and defined over a manifold, which can be approximated by a regular simplicial complex.

Chapter 4

Applications and Concluding Remarks

Geometric modeling is an evolving field with important applications in engineering design, analysis and production of complex engineering objects, vehicles and structures. Furthermore, geometric modeling can be applied in representation and interrogation of large data bases for physical properties (e.g. geophysical, ocean databases), in computer visualization of physical phenomena and in Geographical Information Systems (GIS). However, the application of generalized boundary representations in databases and visualization of physical phenomena is an active research topic. Generalized boundary representations are capable in addressing all above problems, because of their modeling power and generality. They are based on sound mathematical principles and computer science concepts. As we will see below, they are able to cover the major product development stages at varied levels of abstraction ranging from low-level representation to modeling and user interfaces. Despite high computational complexity of geometric modeling algorithms, current computer hardware and software technologies permit the development of efficient applications for large and complex systems.

4.1 Geometric Modeling for Computer-Aided Engineering

Table 4.1 conceptualizes the application of geometric modeling in Computer-Aided Engineering. In particular, we are concerned with the full range of product development stages, from preliminary design to production and with the levels of abstraction necessary for implementation of geometric modeling systems and their use by practitioners.

The horizontal axis of Table 4.1 involves the stages of:

- Design, in which the geometric shape of a product is defined in detail;
- Analysis, in which geometric models are subjected to performance and manufacturing evaluation and simulation;
- Production, in which a physical realization of the object is constructed or maintained later in the life-cycle of the product.

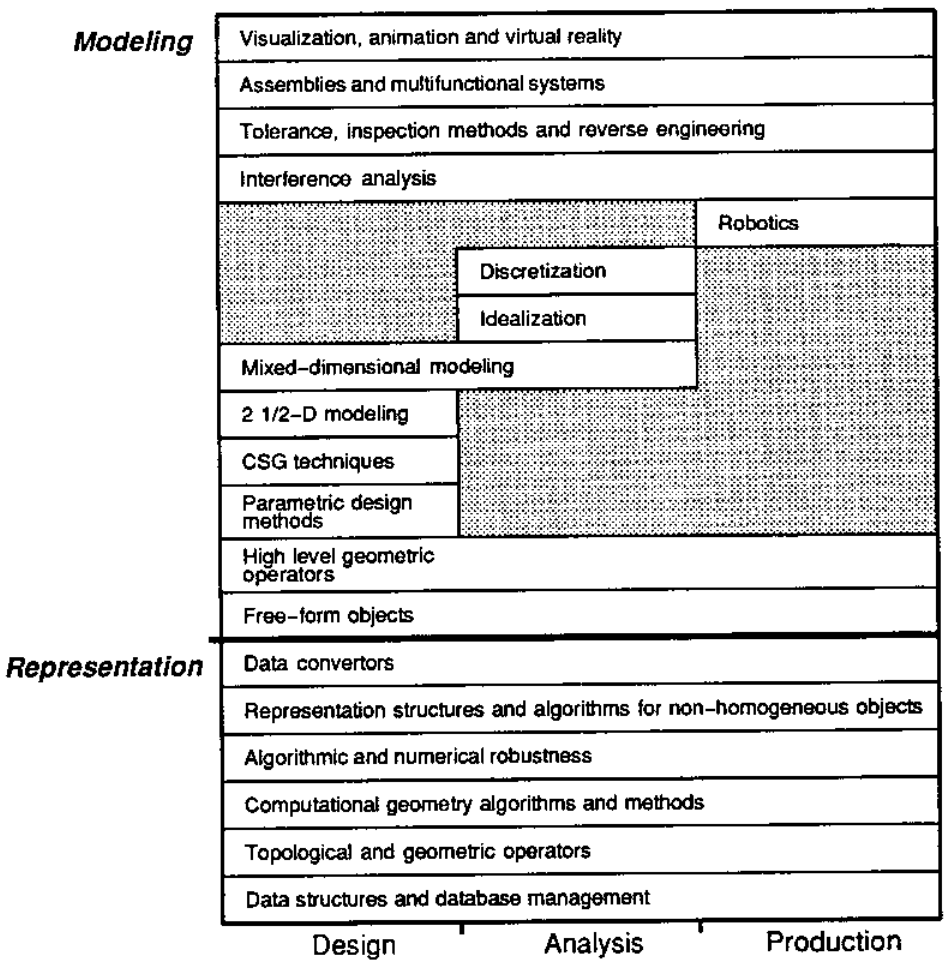

Table 4.1: Product Development Stages

It should be emphasized that feedback from analysis and production to design is an essential part of the product development process.

The vertical axis of Table 4.1, which identifies the abstraction level of a modeling activity, is subdivided into the following layers.

- Representation layer, in which all data structures and schemas and their basic low-level manipulation algorithms reside.
- Modeling layer, through which designers, analysts and manufacturing engineers define, interact and interrogate the geometric model of the object. This layer provides the tools and algorithms for converting user input to structures used by the lowest layer and, conversely, for interrogating the lowest layer in response to user queries.

In the following, we discuss each layer of abstraction in Table 4.1.

4.1.1 Representation Layer

The following problems and issues pertain to the representation layer.

1. **Data structures and database management.** Chapter 3 of this monograph describes a variety of data structures for generalized boundary representations of geometric objects. Modern object-oriented techniques permit the effective implementation of these data structures and basic algorithms for their manipulation. Relational and object-oriented database management systems have been frequently employed for data organization.

2. **Topological and geometric operators.** Topological operators include the generalized Euler operators and data structure interrogation operators. These operators permit the creation and modification of objects at the lowest level and the extraction of incidence and adjacency information. Geometric operators involve primarily the solution of systems of linear and non-linear equations. These systems arise in the computation of intersections of geometric entities, distance computations, etc. [9, 101, 120].

3. **Computational geometry algorithms and methods.** These include primarily the solution of combinatorial problems in geometry, such as point classification, convex hulls, Voronoi diagrams and medial axis transforms [85, 40].

4. **Algorithmic and numerical robustness.** Algorithmic robustness is an important consideration in the solution of non-linear problems, such as intersections of free-form surfaces. Numerical robustness is related to the behavior of exact algorithms when implemented in finite precision [50]. Robustness problems in the presence of imprecise arithmetic are an important issue for basic research. Interval arithmetic methods offer the potential for numerically reliable (verifiable) computation and are a topic of active research in computational geometry [9, 68, 67, 76].

5. **Representation structures and algorithms for non-homogeneous objects.** The advent of composites and Solid Free-Form Fabrication (SFF) methods permit the manufacture of complex objects with non-homogeneous internal properties. Modeling of structures which permit the definition of multi-cell objects naturally permit the representation and interrogation of non-homogeneous objects with piecewise constant internal properties. More complex structures are needed for representing continuously varying internal properties and more research is needed in adequately addressing this problem.

6. **Data convertors.** Effective communication between two CAD systems with different mathematical formulations is a prerequisite for realizing the potential of CAE. Data convertors are algorithms which translate the internal representation of an object from system to system. Important issues are preservation of information content in the translation process, numerical accuracy, efficiency and computer memory usage [5, 52, 79].

4.1.2 Modeling Layer

Within the modeling layer we distinguish primarily design, interrogation and manufacturing issues. However, some issues cut across the traditional subdivision of design, analysis and production and will be dealt with under the term general modeling tools and facilities.

Design issues and methods include:

1. **The definition and interrogation of objects with a free-form boundary.** Piecewise rational polynomial curves and surfaces, for example the Non-Uniform Rational B-Splines (NURBS) are the most popular representation. However, their use has important consequences for algorithm robustness and efficiency. Design methods with free-form surfaces include approximation and interpolation techniques, interactive tweaking and bending operators.

2. **The definition of objects with high level geometric operators, such as offsets, sweeps and blends.** The introduction of these additional surface types has serious consequences on algorithm efficiency and robustness [82, 6, 38].

3. **Parametric design methods.** These methods assist the user in defining and modifying complex shapes of a particular topology but with dimensions obeying particular constraints.

4. **CSG techniques** permit the definition of solid objects using Boolean operators on simple primitives. They facilitate the definition of complex mechanical parts through the processes of material positioning, removal and addition. In most current modelers, CSG is one of the more important methods of shape specification. CSG objects carry with them a procedural history of object definition and are, therefore, useful in editing operations. CSG models are typically converted to boundary representation models for further processing, such as visualization.

5. **$2\frac{1}{2}$-D geometric modeling.** This style of modeling method is used extensively in architectural applications, particularly in the early design phases. They reflect the procedures used by practitioners and are effective, because they abstract and simplify the complex objects designed within this domain. $2\frac{1}{2}$-D models are converted downstream to 3-D models for more refined levels of interrogation.

6. **Mixed-dimensional modeling.** In the design of complex vehicles and structures, the most common approach involves design specification through a sequence of entities of different dimensionality. Typically, designs start with wireframe models, which provide an outline of the shape. These models are next skinned with surfaces to provide a more detailed external shape definition. Finally the surface model is offset to create thin plates and shells. These now enclose architectural spaces, which are further subdivided in functional subspaces, which are then equipped with smaller subassemblies and objects. Ships are frequently designed with this type of process and therefore require the availability of modelers, which can support mixed-dimensional objects.

Interrogation methods include:

1. **Idealization** of objects for more efficient analysis. This is particularly important for complex systems, such as buildings, ships and airplanes. For example, idealization methods include the reduction of dimensionality of objects for effective analysis, such as a three-dimensional beam model to represent a building or a ship, surfaces to represent injection molded parts or thin plates and shells, etc. Skeleton calculation methods provide one of the more useful techniques for dimensional reduction of simple parts [41, 83, 104].

2. **Discretization methods** involve meshing or gridding for finite element, boundary element and computational fluid dynamic analysis. Automating the meshing process is a topic of current research and development. A special issue of *Advances in Engineering Software and Workstations* edited by Armstrong [3] provides an overview of the state of the art in this area.

   An important manufacturing issue is the planning of robotic operations. The field draws upon geometric modeling and computational geometry, combinatorial optimization, higher than three-dimensional modeling (to account for a large number of degrees of freedom), and operations research methods. As an example we refer to the special topic of planning robotic operations for pocket machining [47].

Under the term general modeling tools and facilities we include the following:

1. **Interference analysis** involving, in general, a large number of fixed or moving objects. Such analysis is used in design specification and verification and in the planning of manufacturing and robotic operations [54, 87, 102].

2. **Tolerances and inspection methods.** In order to bound the possible uncertainty of a manufacturing process, designers specify tolerances which provide a range for acceptable performance, function and assembly. Inspection is the process of verifying that a manufactured product has an inaccuracy satisfying the tolerance constraints. This topic is also related to reverse engineering methods, in which complete geometric models of manufactured or physical objects are generated using measurements [21, 51].

3. **Assemblies and multifunctional systems.** Complex objects require the definition of assemblies of subsystems, which in turn may be composed of assemblies of other subsystems. Graph theoretic methods can be used conveniently for this purpose.

4. **Feature representation and recognition.** Feature representation and feature recognition are needed in design, analysis and manufacture and remains an important outstanding problem [78, 93, 106, 107].

5. **Visualization, animation and virtual reality methods.** Visual verification of a design is to this day one of the most fundamental steps of the design process. Visualization techniques, an early objective of computer graphics, has developed into a field of its own with professional conferences and international events [55, 81, 80, 92]. Animation permits the visualization and simulation of moving and deforming objects and has also developed into an active field of research [105]. Finally, virtual reality methods promise to transform the two-dimensional world of computer graphics visualization into a three-dimensional interactive medium which includes other sensory information. The field is still in its infancy but rapidly evolving [32, 113].

4.2 Generalized Boundary-Representations for Computer-Aided Engineering

First generation systems of the seventies were based on the use of Euler operators to guarantee the representation of valid manifold objects. Many of the early commercial CAD systems were based on the above formulation but were accessed with more intuitive user interfaces. For example, higher level algorithms or CSG-like interfaces are necessary for modelers of this kind to be of a greater practical value.

The first generation modelers were applied in visualization, robotic planning (NC machining and tool path generation), interference analysis and in semi-automated methods of object discretization. Most early experimental modelers from this class used linear faceted models, primarily for reasons of efficiency and simplicity of implementation of geometric algorithms. In the intervening time, the Computer-Aided Geometric Design (CAGD) community has perfected the theory and algorithms for generating and processing free-form curves and surfaces. More recently, commercial systems which use the topological structures of these early models but support free-form geometries have emerged. These systems have greatly expanded the domain of application of geometric modeling techniques to support the design of complex objects. However, these systems have many weaknesses because of algorithmic and numerical robustness problems that remain unresolved to this day. Another weakness of the first generation boundary representation systems was that Boolean set operations (a most common technique in solid modeling) were not closed, namely non-manifold objects resulted from their application. This led to a new generation of systems.

The second generation of geometric modelers, developed in the late eighties, permit the explicit representation of mixed-dimensional and non-manifold geometric objects with internal structures and boundaries. These modelers are still of an experimental nature, but hold the promise of a greatly enhanced range of applications. For example, they permit a systematic use of geometric modeling in preliminary design, when incomplete and mixed-dimension models are only available. Similarly, they permit the representation of idealizations and of functional and manufacturing features. Overall, the second generation systems can support a greater variety of objects and applications at the expense of more complex data structures. Generalized non-manifold and mixed-dimensional data structures can represent wireframe, surface and solid models in a single, unified environment. Unlike the early Euler-based systems, the second generation systems have deeper differences among themselves.

On a path parallel to the development of non-manifold and mixed dimensional models, computer scientists have devised abstract modeling concepts with the objective of supporting computational geometry algorithms, e.g. Voronoi diagrams and Delaunay triangulations. Abstract models are based on elegant mathematical structures and provide a small number of functions for incremental construction and modification of objects. Except for the representation of mixed-dimensional and non-manifold objects, abstract modeling concepts permit the representation of n-dimensional multi-cell objects with internal boundaries much in the same way as the second generation modelers. For this reason, it is a historical curiosity that abstract modeling concepts and the second generation geometric modelers were developed largely independently.

Bibliography

[1] M. K. Agoston. *Algebraic Topology.* Marcel Dekker Inc., New York, 1976.

[2] S. Ansaldi, L. De Floriani, and B. Falcidieno. Geometric modeling of solid objects by using a face adjacency graph representation. *Proceedings of SIGGRAPH '85, Computer Graphics*, 19(3):131–140, July 1985.

[3] C. G. Armstrong, editor. *Advances in Engineering Software and Workstations.* 13(5/6), September/November 1991.

[4] M. A. Armstrong. *Basic Topology.* Springer-Verlag, New York, 1983.

[5] L. Bardis and N. M. Patrikalakis. Approximate conversion of rational B-spline patches. *Computer Aided Geometric Design*, 6(3):189–204, August 1989.

[6] L. Bardis and N. M. Patrikalakis. Surface approximation with rational B-splines. *Engineering with Computers*, 6(4):223–235, 1990.

[7] B. Baumgart. A polyhedron representation for computer vision. In *National Computer Conference*, pages 589–596. AFIPS Conference Proceedings, 1975.

[8] C. Berge. *Graphs and Hypergraphs.* North Holland, 1973. Translated and revised edition of 'Graphes et Hypergraphes', Dunod, Paris, 1970.

[9] C. Bliek. *Computer Methods for Design Automation.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, July 1992.

[10] A. Bowyer and J. Woodwark. *A Programmer's Geometry.* Butterworth and Co., Ltd., Sevenoaks, England, 1983.

[11] I. C. Braid, R. C. Hillyard, and I. A. Stroud. Stepwise construction of polyhedra in geometric modeling. In K. W. Brodlie, editor, *Mathematical Methods in Computer Graphics and Design*, pages 123–141, Academic Press, London, 1980.

[12] E. Brisson. Representing geometric structures in d dimensions: Topology and order. In *Proceedings of the Fifth ACM Symposium on Computational Geometry*, pages 218–227, Saarbrücken, Germany, June 1989.

[13] E. Brisson. *Representation of d Dimensional Geometric Objects.* PhD thesis, Department of Computer Science and Engineering, University of Washington, Seattle, 1990.

[14] E. Brisson. Representing geometric structures in d dimensions: Topology and order. *Discrete and Computational Geometry*, 9:387–426, 1993.

[15] C. M. Brown. PADL-2: A technical summary. *IEEE Computer Graphics and Applications*, 2(2):69–84, March 1982.

[16] P. Brunet and D. Ayala. Extended octtree representation of free-form surfaces. *Computer Aided Geometric Design*, 4(1-2):141–154, 1987.

[17] C. E. Buckley. A divide-and-conquer algorithm for computing 4-dimensional convex hulls. In H. Noltemeier, editor, *Computational Geometry and its Applications*, pages 113–135. Proceedings of the International Workshop on Computational Geometry, Springer-Verlag, 1988.

[18] M. S. Casale. Free-form solid modeling with trimmed surface patches. *IEEE Computer Graphics and Applications*, 7(1):33–43, January 1987.

[19] J.-M. Chen. *Integration of Parametric Geometry and Non-Manifold Topology in Geometric Modeling.* PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, April 1993.

[20] J.-M. Chen, E. L. Gürsöz, and F. B. Prinz. Integration of parametric geometry and non-manifold topology in geometric modeling. In J. Rossignac, J. Turner, and G. Allen, editors, *Proceedings of the Second Symposium on Solid Modeling and Applications*, pages 53–64, Montreal, Canada, May 1993. ACM SIGGRAPH in cooperation with the IEEE Computer Society.

[21] P. N. Chivate and A. G. Jablokow. Solid-model generation from measured point data. *Computer-Aided Design*, 25(9):587–600, September 1993.

[22] H. Chiyokura. An extended rounding operator for modeling solids with free-form surfaces. In T. L. Kunii, editor, *Proceedings of Computer Graphics International '87*, pages 249–268. Springer-Verlag, 1987.

[23] H. Chiyokura. *Solid Modelling with DESIGNBASE: Theory and Implementation.* Addison-Wesley, 1988.

[24] H. Chiyokura and F. Kimura. Design of solids with free-form surfaces. In *Proceedings of SIGGRAPH '83, Computer Graphics 17(3)*, pages 289–298, July 1983.

[25] Y. Choi. *Vertex-Based Boundary Representation of Non-Manifold Geometric Models.* PhD thesis, Carnegie-Mellon University, Pittsburgh, PA, August 1989.

[26] C. Chryssostomidis and N. M. Patrikalakis. Geometric modeling issues in computer aided design of marine structures. *Marine Technology Society Journal*, 22(2):15–33, December 1988.

[27] L. De Floriani and B. Falcidieno. A hierarchical boundary model for solid object representation. *ACM Transactions on Graphics*, 7(1):42–60, January 1988.

[28] H. Desaulniers and N. F. Stewart. An extension of manifold boundary representations to the r-sets. *ACM Transactions on Graphics*, 11(1):42–60, January 1992.

[29] D. P. Dobkin and M. J. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. In *Proceedings of the Third ACM Symposium on Computational Geometry*, pages 86–99, Waterloo, Canada, June 1987.

[30] D. P. Dobkin and M. J. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. *Algorithmica*, 4:3–32, 1989.

[31] J. F. Dufourd. Formal specification of topological subdivisions using hypermaps. *Computer Aided Design*, 23(2):99–116, February 1991.

[32] R. Earnshaw, M. Gigante, and H. Jones, editors. *Virtual Reality Systems*. Academic Press, London, 1993.

[33] C. Eastman and M. Henrion. GLIDE: A language for design information systems. *Computer Graphics*, 11(2):24–33, July 1977.

[34] C. Eastman and R. Thornton. A report on the GLIDE2 language definition. Technical report, CAD Group, Institute of Physical Planing, Carnegie-Mellon University, Pittsburgh, PA, March 1979.

[35] C. Eastman and K. Weiler. Geometric modeling using Euler operators. In *Proceedings of the First Annual Conference on Computer Graphics in CAD/CAM Systems*, pages 248–259, May 1979.

[36] J. Edmonds. A combinatorial representation for polyhedral surfaces. *American Mathematical Society Notices*, 7:646, October 1960.

[37] R. T. Farouki. Trimmed surface algorithms for the evaluation and interrogation of solid boundary representations. *IBM Journal of Research and Development*, 31(3):314–334, May 1987.

[38] P. C. Filkins, S. T. Tuohy, and N. M. Patrikalakis. Computational methods for blending surface approximation. *Engineering with Computers*, 9(1):49–61, 1993.

[39] P. J. Giblin. *Graphs, Surfaces and Homology*. Chapman and Hall, 1981.

[40] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, April 1985.

[41] H. N. Gursoy and N. M. Patrikalakis. Automated interrogation and adaptive subdivision of shape using medial axis transform. *Advances in Engineering Software and Workstations*, 13(5/6):287–302, September/November 1991.

[42] E. L. Gursoz and Y. Choi. *NOODLES User Manual*. CAD Group, Carnegie-Mellon University, Pittsburgh, PA, 1991.

[43] E. L. Gürsöz, Y. Choi, and B. F. Prinz. Vertex-based representation of non-manifold boundaries. In M. J. Wozny, J. U. Turner, and K. Preiss, editors, *Geometric Modeling for Product Engineering*, pages 107–130, Elsevier Science Publishers, Holland, 1990.

[44] E. L. Gürsöz, Y. Choi, and F. B. Prinz. Boolean set operations in non-manifold representation objects. *Computer Aided Design*, 23(1):33–39, January 1991.

[45] F. Harary. *Graph Theory*. Addison-Wesley, Reading, MA, 1969.

[46] L.-X. He. A non-manifold geometry modeler: An object oriented approach. Master's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, February 1993.

[47] M. Held. *On the Computational Geometry of Pocket Machining*. Springer-Verlag, Berlin, Germany, 1991.

[48] M. Henle. *A Combinatorial Introduction to Topology*. W. H. Freeman and Company, San Francisco, 1979.

[49] C. M. Hoffmann. *Geometric and Solid Modeling: An Introduction*. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1989.

[50] C. M. Hoffmann. The problems of accuracy and robustness in geometric computation. *Computer*, 22(3):31–41, March 1989.

[51] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of SIGGRAPH '92, Computer Graphics, 26(2)*, pages 71–78. ACM, July 1992.

[52] J. Hoschek and F.-J. Schneider. Approximate spline conversion for integral and rational Bézier and B-Spline surfaces. In R. E. Barnhill, editor, *Geometry Processing for Design and Manufacturing*, pages 45–87. SIAM, Philadelphia, 1992.

[53] G. M. Hunter and K. Steiglitz. Operations on images using quad trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):145–153, 1979.

[54] R. A. Jinkerson, S. L. Abrams, L. Bardis, C. Chryssostomidis, A. Clément, N. M. Patrikalakis, and F.-E. Wolter. Inspection and feature extraction of marine propellers. *Journal of Ship Production*, 9(2):88–106, May 1993.

[55] A. Kaufman. Introduction to volume synthesis. In N. M. Patrikalakis, editor, *Scientific Visualization of Physical Phenomena*, pages 25–36. Tokyo: Springer-Verlag, 1991.

[56] F. Kimura. Geomap-III: Designing solids with free-form surfaces. *IEEE Computer Graphics and Applications*, 4(6):58–72, June 1984.

[57] G. A. Kriezis. *Algorithms for Rational Spline Surface Intersections*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, March 1990.

[58] G. A. Kriezis and N. M. Patrikalakis. Rational polynomial surface intersections. In G. A. Gabriele, editor, *Proceedings of the 17th ASME Design Automation Conference, Vol. II*, pages 43–53, Miami, September 1991. ASME, New York, 1991.

[59] G. A. Kriezis, N. M. Patrikalakis, and F.-E. Wolter. Topological and differential equation methods for surface intersections. *Computer Aided Design*, 24(1):41–55, January 1992.

[60] G. A. Kriezis, P. V. Prakash, and N. M. Patrikalakis. A method for intersecting algebraic surfaces with rational polynomial patches. *Computer Aided Design*, 22(10):645–654, December 1990.

[61] T. L. Kunii, T. Sato, and K. Yamaguchi. Generation of topological boundary representations from octtree encoding. *IEEE Computer Graphics and Applications*, 5(3):29–38, March 1985.

[62] M. J. Laszlo. *A Data Structure for Manipulating Three-Dimensional Subdivisions*. PhD thesis, Department of Computer Science, Princeton University, August 1987.

[63] Y. T. Lee and A. A. G. Requicha. Algorithms for computing the volume and other integral properties of solid objects, I: Known methods and open issues. *Communications of the ACM*, 25(9):635–641, September 1982.

[64] Y. T. Lee and A. A. G. Requicha. Algorithms for computing the volume and other integral properties of solids. II: A family of algorithms based on representation conversion and cellular approximation. *Communications of the ACM*, 25(9):642–650, September 1982.

[65] P. Lienhardt. Subdivisions of n-dimensional spaces and n-dimensional generalized maps. In *Proceedings of the Fifth ACM Symposium on Computational Geometry*, pages 228–236, Saarbruecken, Germany, June 1989.

[66] P. Lienhardt. Topological models for boundary representation: A comparison with n-dimensional generalized maps. *Computer Aided Design*, 23(1):59–82, January 1991.

[67] T. Maekawa and N. M. Patrikalakis. Computation of singularities and intersections of offsets of planar curves. *Computer Aided Geometric Design*, 10(5):407–429, October 1993.

[68] T. Maekawa and N. M. Patrikalakis. Interrogation of differential geometry properties for design and manufacture. *The Visual Computer*, 10(4):216–237, March 1994.

[69] M. Mäntylä. *An Introduction to Solid Modeling*. Computer Science Press, Rockville, Maryland, 1988.

[70] M. Mäntylä and R. Sulonen. GWB - A solid modeller with Euler operators. *IEEE Computer Graphics and Applications*, 2(7):17–32, September 1982.

[71] W. S. Massey. *A Basic Course in Algebraic Topology*. Springer-Verlag, 1991.

[72] H. Masuda. Topological operators and Boolean operations for complex-based nonmanifold geometric models. *Computer Aided Design*, 25(2):119–129, February 1993.

[73] D. Meagher. Geometric modeling using octtree encoding. *Computer Graphics and Image Processing*, 19:129–147, June 1982.

[74] J. R. Munkres. *Topology: a First Course*. Prentice-Hall, Englewood Cliffs, NJ, 1975.

[75] J. R. Munkres. *Elements of Algebraic Topology*. Addison-Wesley, 1984.

[76] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, 1990.

[77] A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-independent modeling with simplicial complexes. *ACM Transactions on Graphics*, 12(1):56–102, January 1993.

[78] S. Parry-Barwock and A. Bowyer. Is the features interface ready? In R. Martin, editor, *Directions in Geometric Computing*, pages 129–160. Information Geometers, Winchester, UK, 1993.

[79] N. M. Patrikalakis. Approximate conversion of rational splines. *Computer Aided Geometric Design*, 6(2):155–165, 1989.

[80] N. M. Patrikalakis, editor. *Scientific Visualization of Physical Phenomena, Proceedings of the 9th International Conference on Computer Graphics, CGI '91, MIT, Cambridge, MA, June 1991*, Tokyo, 1991. Springer.

[81] N. M. Patrikalakis, editor. *Visualization in Science and Engineering, Special Issue of The Visual Computer from CGI '91, Vol. 8, Nos. 5-6.* Springer, June 1992.

[82] N. M. Patrikalakis. Surface-to-surface intersections. *IEEE Computer Graphics and Applications*, 13(1):89–95, January 1993.

[83] N. M. Patrikalakis and L. Bardis. Feature extraction from B-spline marine propeller representations. *Journal of Ship Research*, 36(3):233–247, September 1992.

[84] N. M. Patrikalakis and P. V. Prakash. Surface intersections for geometric modeling. *Journal of Mechanical Design, ASME Transactions*, 112(1):100–107, March 1990.

[85] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, 1985.

[86] A. A. G. Requicha. Representations of solid objects - theory, methods and systems. *ACM Computing Surveys*, 12(4):437–464, December 1980.

[87] A. A. G. Requicha. Progress in solid modeling and its applications. In *Proceedings of the 18th NSF Design and Manufacturing Systems Conference*, pages 761–766, Atlanta, January 1992. SME.

[88] A. A. G. Requicha. Solid modeling - a 1988 update. In B. Ravani, editor, *CAD-Based Programming for Sensory Robots*, pages 3–22, New York: Springer-Verlag, 1988.

[89] A. A. G. Requicha and H. B. Voelcker. Boolean operations in solid modeling: Boundary evaluation and merging algorithms. *Proceedings of the IEEE*, 3(7):30–44, October 1983.

[90] A. A. G. Requicha and H. B. Voelcker. Solid modeling: Current status and research directions. *IEEE Computer Graphics and Applications*, 3(7):25–37, October 1983.

[91] A. A.G. Requicha and J. R. Rossignac. Solid modeling and beyond. *IEEE Computer Graphics and Applications*, 12(5):31–44, September 1992.

[92] D. F. Rogers and R. A. Earnshaw, editors. *State of the Art in Computer Graphics.* Springer-Verlag, New York, 1991.

[93] J. R. Rossignac. Issues on feature-based editing and interrogation of solid models. *Computers and Graphics*, 14(2):149–172, 1990.

[94] J. R. Rossignac and M. A. O'Connor. SGC: A dimension-independent model for pointsets with internal structures and incomplete boundaries. In M. J. Wozny, J. U. Turner, and K. Preiss, editors, *Geometric Modelling for Product Engineering*, pages 145–180, Holland, Elsevier Science Publishers, 1990.

[95] J. R. Rossignac and A. G. Requicha. Constructive non-regularized geometry. *Computer Aided Design*, 23(1):21–32, January 1991.

[96] S. D. Roth. Ray casting for modeling solids. *Computer Graphics and Image Processing*, 18(2):109–144, 1982.

[97] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 15:187–260, 1984.

[98] H. Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS.* Addison-Wesley, Reading, MA, 1990.

[99] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading, MA, 1990.

[100] H. Samet and R. E. Webber. Data structures to support Bézier-based modeling. *Computer Aided Design*, 23(3):162–176, April 1991.

[101] E. C. Sherbrooke and N. M. Patrikalakis. Computation of the solutions of nonlinear polynomial systems. *Computer Aided Geometric Design*, 10(5):379–405, October 1993.

[102] A. J. Spyridi and A. A. G. Requicha. Accessibility analysis for polyhedral objects. In S. G. Tzafestas, editor, *Engineering Systems with Intelligence: Concepts, Tools and Applications*, pages 317–324. Kluwer Academic Publishers, 1991.

[103] R. Stillwell. *Classical Topology and Combinatorial Group Theory. Number 72 in Graduate Texts in Mathematics.* Springer-Verlag, 1984.

[104] A. Sudhalkar, L. Gürsöz, and F. Prinz. Continuous skeletons of discrete objects. In J. Rossignac, J. Turner, and G. Allen, editors, *Proceedings of the Second Symposium on Solid Modeling and Applications*, pages 85–94. ACM, 1993.

[105] N. M. Thalmann and D. Thalmann, editors. *Journal of Visualization and Computer Animation.* 1990.

[106] J. H. Vandebrande and A. A. G. Requicha. Spatial reasoning for the automatic recognition of interacting form features. In *Proceedings of the ASME International Conference on Computers in Engineering*, volume 1, pages 251–256, Boston, August 1990. ASME.

[107] J. H. Vandebrande and A. A. G. Requicha. Spatial reasoning for the automatic recognition of machinable features in solid models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1993. To appear.

[108] K. Weiler. Edge-based data structures for solid modeling in curved surface environments. *IEEE Computer Graphics and Applications*, 5(1):21–40, January 1985.

[109] K. Weiler. Boundary graph operators for non-manifold geometric modeling representations. In M. J. Wozny, H. McLaughlin, and J. Encarnacao, editors, *Geometric Modeling for CAD Applications*, pages 37–66, Elsevier Science Publishers, Holland, 1986.

[110] K. Weiler. The radial edge structure: A topological representation for non-manifold geometric modeling. In M. J. Wozny, H. McLaughlin, and J. Encarnacao, editors, *Geometric Modeling for CAD Applications*, pages 3–36, Elsevier Science Publishers, Holland, 1986.

[111] K. Weiler and D. McLachlan. Generalized sweep operators in the non-manifold environment. In M. J. Wozny, J. U. Turner, and K. Preiss, editors, *Geometric Modeling for Product Engineering*, pages 87–106, Elsevier Science Publishers, Holland, 1990.

[112] K. J. Weiler. *Topological Structures for Geometric Modeling.* PhD thesis, Rensselaer Polytechnic Institute, Troy, NY, 1986.

[113] A. Wexelblat, editor. *Virtual Reality: Applications and Explorations.* Academic Press, London, 1993.

[114] P. R. Wilson. Euler formulas and geometric modeling. *IEEE Computer Graphics and Applications*, 5(8):45–60, August 1985.

[115] P. R. Wilson. Solid modeling R&D in the USA. In *European Conference on Solid Modeling*, London, September 1985.

[116] A. Wong and D. Sriram. Geometric modeling for cooperative product development. In J. Rossignac, J. Turner, and G. Allen, editors, *Proceedings of the Second Symposium on Solid Modeling and Applications*, pages 497–498. ACM SIGGRAPH in cooperation with the IEEE Computer Society, May 1993.

[117] T. C. Woo. A combinatorial analysis of boundary data structure schemata. *IEEE Computer Graphics and Applications*, 5(3):38–52, March 1985.

[118] T. C. Woo and J. D. Wolter. A constant expected time, linear storage data structure for representing three-dimensional objects. *IEEE Transactions on Systems, Man, and Cybernetics*, 14(3):510–515, May/June 1984.

[119] K. Yamaguchi, T. L. Kunii, K. Fujumura, and H. Toriya. Octtree-related data structures and algorithms. *IEEE Computer Graphics and Applications*, 4(6):24–37, June 1984.

[120] J. Zhou, E. C. Sherbrooke, and N. M. Patrikalakis. Computation of stationary points of distance functions. *Engineering with Computers*, 9(4):231–246, Winter 1993.

Index